

ODRecord

Class Definition File: ODRECORD.IDL

Class Hierarchy

SOMObject
 ODObject
 ODDesc
 ODDescList
 ODRecord

Description

An object of the ODRecord class is a wrapper for an AE record (type [AERecord](#)), a descriptor list that can be used to construct OSA event parameters.

An AE record is a special descriptor list that allows keyword-specified [descriptor](#) records for OSA event parameters.

For more information on OSA events and the [AERecord](#) type, see the chapter introducing OSA events in the *Open Scripting Architecture Guide and Reference for OS/2* . For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

Methods

The methods defined by the ODRecord class include:

- [InitODRecord](#)

Overridden Methods

There are currently no methods overridden by the ODRecord class.

InitODRecord

InitODRecord - Syntax

This method initializes this AE record.

```
#define INCL_ODRECORD
#define INCL_ODAPI
#include <os2.h>
```

```
InitODRecord();
```

InitODRecord - Return Value

None.

InitODRecord - Parameters

None.

InitODRecord - Remarks

There is no factory method for the [ODRecord](#) class; after creating a new AE record, OpenDoc or your part must call this method to initialize a new AE record.

InitODRecord - Topics

Class:
ODRecord

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

ODRefCntObject

Class Definition File: REFCTOBJ.IDL

Class Hierarchy
SOMObject
 ODObject
 ODRefCntObject

Description

An object of the ODRefCntObject class implements reference counting, a mechanism that allows OpenDoc to manage memory used by objects.

In a typical OpenDoc session, various objects are shared by other objects, each of which has a reference to the shared object. A shared object should not be deleted as long as any object is using it. OpenDoc uses *reference counting* to manage memory for shared objects. Each reference-counted object keeps track of the number of existing references to it. An object's *reference count* indicates the number of objects that are currently using it. When an object's reference count drops to 0, no other object is using it; only then is it safe to delete the object, freeing the space it occupies in memory.

The ODRefCntObject class is the abstract superclass for all OpenDoc classes whose objects require reference counting. You should never instantiate ODRefCntObject itself, but you can instantiate a subclass by calling the appropriate factory method. If the factory method creates a new object, it sets the reference count for that object to 1; if the factory method returns a reference to an existing object, it increments that object's reference count by 1.

Memory management for each reference-counted class is the responsibility of its factory class. For example, the [ODStorageSystem](#) class is the factory class for the [ODContainer](#) class; that is, the storage system's factory method creates a container object. When the container object's reference count drops to 0, it informs the storage system, which deletes the container object.

Methods

The methods defined by the ODRefCntObject class include:

- [Acquire](#)
- [GetRefCount](#)
- [InitRefCountObject](#)
- [Release](#)

Overridden Methods

There are currently no methods overridden by the ODRefCntObject class.

Acquire

Acquire - Syntax

This method increments an object's reference count by 1.

```
#define INCL_ODREFCNTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
Acquire();
```

Acquire - Return Value

None.

Acquire - Parameters

None.

Acquire - Remarks

Most methods that return a reference to a reference-counted object increment the object's reference count; however, if your part obtains a reference to a reference-counted object from a method that does not increment the object's reference count, you should call the object's [Acquire](#) method before you cache the reference in any structure. When the reference is replaced or removed from the data structure, you should call the object's [Release](#) method to decrement its reference count.

Acquire - Override Policy

If you subclass [ODRefCntObject](#), you must override this method. If you subclass [ODPart](#), you must override this method if your part performs any specific actions when its reference count is incremented. Your override method must call its inherited method at the beginning of its implementation.

Acquire - Topics

Class:

[ODRefCntObject](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

GetRefCount

GetRefCount - Syntax

This method returns the current reference count of this object.

```
#define INCL_ODREFCNTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetRefCount();
```

GetRefCount Return Value - rv

rv ([ODULong](#)) - returns

The current reference count of this object.

GetRefCount - Parameters

rv ([ODULong](#)) - returns

The current reference count of this object.

GetRefCount - Topics

Class:

ODRefCountObject

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InitRefCountObject

InitRefCountObject - Syntax

This method initializes this object and sets its reference count to 1.

```
#define INCL_ODREFCNTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
InitRefCountObject();
```

InitRefCountObject - Return Value

None.

InitRefCountObject - Parameters

None.

InitRefCountObject - Remarks

This method is not called directly to initialize a reference-counted object, but is called by a subclass-specific initialization method. By

convention, every subclass of [ODRefCntObject](#) should have a separate initialization method (for example, the `InitMyRefCntObject` method) that is called when an instance of that subclass is created, and that may have additional parameters beyond those of the `InitRefCntObject` method. The `InitMyRefCntObject` method should call the inherited `InitRefCntObject` `InitRefCntObject` method at the beginning of its implementation.

InitRefCntObject - Topics

Class:

`ODRefCntObject`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

Release

Release - Syntax

This method decrements an object's reference count by 1,

```
#define INCL_ODREFCNTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
Release();
```

Release - Return Value

None.

Release - Parameters

None.

Release - Remarks

When you no longer need an object reference that you obtained by calling a factory method (for example, the draft's [CreatePart](#) or [AcquirePart](#) method), you should call the object's Release method. In addition, you should balance every call to the object's [Acquire](#) method with a call to its Release method.

Release - Exception Handling

kODErrZeroRefCount

The reference count cannot be decremented because the reference count is already 0.

Release - Override Policy

If you subclass [ODRefCntObject](#), you must override this method to tell the draft to release the object from memory if the object's count becomes 0. If you subclass [ODPart](#), you must override this method. Your override method must call its inherited method at the beginning of its implementation.

Release - Topics

Class:

ODRefCntObject

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

ODSemanticInterface

Class Definition File: SEMTINTF.IDL

Class Hierarchy

SOMObject

ODObject

ODRefCntObject

ODExtension

ODBaseSemanticInterface

ODSemanticInterface

Description

An object of the ODSemanticInterface class implements an extension that handles semantic events for you part. It is recommended, but not required, that your part editor supports this extension.

When a document is opened, the session object creates a single semantic interface object for the document shell. All parts of that document share the document shell's semantic interface object; you can obtain a reference to it by calling the session object's

[AcquireShellSemtInterface](#) method.

The methods defined by the `ODSemanticInterface` class parallel handlers and functions defined for OSA events in the *Open Scripting Architecture Guide and Reference for OS/2*. The following information assumes a basic knowledge of handling OSA events and resolving object specifiers.

The `ODSemanticInterface` class is designed to respond to semantic events received by your part. Because OpenDoc is responsible for dispatching semantic events to your part, many of the handlers that would normally be defined by multiple functions using the OSA Event Manager are grouped conceptually in the `ODSemanticInterface` class. For example, the [CallEventHandler](#) method provides a bottleneck that all OSA events sent to your part must go through.

The methods of this class consist of four types: semantic-event handlers, object accessors, object-callback functions, and other handlers. OpenDoc calls the semantic-event handlers of your subclass to handle OSA events intended for your part. The object accessors and object-callback functions are used by OpenDoc to resolve object specifiers for your part. OpenDoc calls other handlers for special purposes. Your implementation of these methods can consist of separate handlers or a single large procedure that handles all of your part's semantic events.

The `ODSemanticInterface` class is an abstract superclass that you must subclass to create your semantic interface. OpenDoc accesses your semantic interface object by calling your part's [AcquireExtension](#) method, which returns a reference to the extension object. The semantic-interface extension type is identified by the constant `kODExtSemanticInterface`. If your part supports this extension, your part subclass must override the [AcquireExtension](#), [HasExtension](#), and [ReleaseExtension](#) methods and provide an appropriate implementation. For more information related to extension objects, see the class description for [ODEExtension](#).

For more information about creating and sending OSA events to other parts, see the [ODMessageInterface](#) class description. For more information related to resolving object specifiers, see the class description of the [ODNameResolver](#) and the chapter on resolving and creating object specifier records in the *Open Scripting Architecture Guide and Reference for OS/2*. For more information related to OSA events and coercion handlers, see the chapter on responding to OSA events *Open Scripting Architecture Guide and Reference for OS/2*. For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide*.

somInit

This method initializes the instance variables in a som object; it is inherited from the `SOMObject` class.

If you subclass `ODSemanticInterface`, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in your semantic interface object. The SOM library calls this method when this semantic interface is created. You must not do anything that might cause this method to fail. This limits you to operations such as setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this semantic interface object's subclass-specific initialization code; see also the method [InitSemanticInterface](#).

somUninit

This method disposes of the storage created for a SOM object; it is inherited from the `SOMObject` class.

If you subclass `ODSemanticInterface`, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for this semantic interface object, including any storage related to additional instances variables initialized in this semantic interface object. The SOM library calls this method when this semantic interface object is deleted; this method must not fail.

Release

This method decrements an object's reference count by 1; it is inherited from the [ODRefCntObject](#) class.

```
void Release ();
```

If you subclass `ODSemanticInterface`, you can override this method if your part needs to release an object and reclaim valuable resources like memory. Your override method must call its inherited method at the beginning of your implementation.

The inherited `Release` method decrements this semantic interface object's reference count by 1. The inherited method may delete this semantic interface object from memory if this object's reference count becomes 0. A part editor calls this method when it no longer needs a reference to this semantic interface object.

Purge

This method frees memory on request; it is inherited from the [ODOObject](#) class.

```
ODSize Purge (ODSize size);
```

If you subclass [ODObject](#), you can override this method and should do so if it creates caches and temporary buffers. If you subclass [ODSemanticInterface](#), you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method because you will need to compute the value returned from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the sum as the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

Methods

The methods defined by the [ODSemanticInterface](#) class include the following, grouped according to purpose:

Initializing

- [InitBaseSemanticInterface](#)
- [InitSemanticInterface](#)

Handling Semantic-Events

- [CallEventHandler](#)
- [InstallEventHandler](#)
- [RemoveEventHandler](#)

Object Accessors

- [CallObjectAccessor](#)
- [InstallObjectAccessor](#)
- [RemoveObjectAccessor](#)

Object-Callback Functions

- [CallAdjustMarksProc](#)
- [CallCompareProc](#)
- [CallCountProc](#)
- [CallDisposeTokenProc](#)
- [CallGetErrDescProc](#)
- [CallGetMarkTokenProc](#)
- [CallMarkProc](#)
- [InstallAdjustMarksProc](#)
- [InstallCoercionHandler](#)
- [InstallCompareProc](#)
- [InstallCountProc](#)
- [InstallDisposeTokenProc](#)
- [InstallGetErrDescProc](#)
- [InstallGetMarkTokenProc](#)
- [InstallMarkProc](#)

Other Handlers

- [CallCoercionHandler](#)
- [CallPredispatchProc](#)
- [InstallSpecialHandler](#)
- [RemoveCoercionHandler](#)
- [RemoveSpecialHandler](#)
- [UsingPredispatchProc](#)

OSL Settings

- [GetOSLSupportFlags](#)
- [SetOSLSupportFlags](#)

Overridden Methods

There are currently no methods overridden by the [ODSemanticInterface](#) class.

CallAdjustMarksProc

CallAdjustMarksProc - Syntax

This method unmarks a series of objects that were previously marked.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart          *thePart;
ODSLong         newStart;
ODSLong         newStop;
ODOSLToken      *markToken;

CallAdjustMarksProc(thePart, newStart, newStop,
                    markToken);
```

CallAdjustMarksProc Parameter - thePart

thePart (ODPart *) - input
A reference to the part associated with this semantic-interface object.

CallAdjustMarksProc Parameter - newStart

newStart (ODSLong) - input
The index that specifies the new beginning of the entries that are to remain marked.

CallAdjustMarksProc Parameter - newStop

newStop (ODSLong) - input
The index that specifies the new end of the entries that are to remain marked.

CallAdjustMarksProc Parameter - markToken

markToken (ODOSLToken *) - input
A reference to the OpenDoc token to be used for unmarking the elements.

CallAdjustMarksProc - Return Value

None.

CallAdjustMarksProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

newStart (ODSLong) - input

The index that specifies the new beginning of the entries that are to remain marked.

newStop (ODSLong) - input

The index that specifies the new end of the entries that are to remain marked.

markToken (ODOSLToken *) - input

A reference to the OpenDoc token to be used for unmarking the elements.

None.

CallAdjustMarksProc - Remarks

OpenDoc calls this method, when it is in the process of resolving an object specifier, to unmark a series of objects that were previously marked by a call to the [CallMarkProc](#) method. The *newStart* and *newStop* parameters indicate the beginning and end, respectively, of the range of marked objects that are to remain marked. Your override of this method should iterate over the remaining objects and unmark them. Your override of this method should use the *markToken* parameter to identify the marked objects in the iteration.

Before OpenDoc can call this method, you must call your semantic-interface object's [SetOSLSupportFlags](#) method and specify the flag `kAEIDoMarking` to indicate that your semantic interface supports marking. In general, your part should override the [CallGetMarkTokenProc](#), [CallMarkProc](#), and [CallAdjustMarksProc](#) methods in the following cases: it already supports the marking of objects or it expects to deal with large numbers of records that might not all fit into memory at once.

Your override of this method is responsible for deallocating any structures for the previously marked objects that were allocated by your [CallMarkProc](#) method. If your part's semantic interface supports the marking of objects, it should also supply a token disposal handler to dispose of your mark token.

CallAdjustMarksProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to adjust the marks as requested.

This method may throw platform-specific exceptions.

CallAdjustMarksProc - Related Methods

Related Methods

- [ODSemanticInterface::CallDisposeTokenProc](#)
- [ODSemanticInterface::CallGetMarkTokenProc](#)
- [ODSemanticInterface::CallMarkProc](#)
- [ODSemanticInterface::SetOSLSupportFlags](#)

CallAdjustMarksProc - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CallCoercionHandler

CallCoercionHandler - Syntax

This method coerces the specified descriptor to a different type.

```
#define INCL_ODSEMANANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *thePart;
ODDesc      *theODDesc;
ODDescType  toType;
ODDesc      *theResult;

CallCoercionHandler(thePart, theODDesc, toType,
                    theResult);
```

CallCoercionHandler Parameter - thePart

thePart (ODPart *) - input
A reference to a part associated with this semantic interface object.

CallCoercionHandler Parameter - theODDesc

theODDesc (ODDesc *) - input
A reference to a descriptor to coerce.

CallCoercionHandler Parameter - toType

toType (ODDescType) - input
A reference to a descriptor type to coerce.

CallCoercionHandler Parameter - theResult

theResult (ODDesc *) - input
A reference to the resulting coerced descriptor.

CallCoercionHandler - Return Value

None.

CallCoercionHandler - Parameters

thePart (ODPart *) - input
A reference to a part associated with this semantic interface object.

theODDesc (ODDesc *) - input
A reference to a descriptor to coerce.

toType (ODDescType) - input
A reference to a descriptor type to coerce.

theResult (ODDesc *) - input
A reference to the resulting coerced descriptor.

None.

CallCoercionHandler - Remarks

OpenDoc does not chain coercion handlers together. For example, an embedded part does not inherit the coercion handler of its containing part, and a part does not inherit the coercion handler of the document shell.

CallCoercionHandler - Exception Handling

The OSA Event Manager may throw an exception if this semantic interface does not support the specified coercion.

This method can throw platform-specific exceptions.

CallCoercionHandler - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

CallCompareProc

CallCompareProc - Syntax

This method compares two descriptors.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODPart          *thePart;
ODDescType      oper;
ODOSLToken      *obj1;
ODOSLToken      *obj2;
ODBoolean       *result;
```

```
CallCompareProc(thePart, oper, obj1, obj2,
                result);
```

CallCompareProc Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

CallCompareProc Parameter - oper

oper (ODDescType) - input
The comparison operator that specifies how to compare the two objects.

CallCompareProc Parameter - obj1

obj1 (ODOSLToken *) - input
A reference to the first object in the comparison.

CallCompareProc Parameter - obj2

obj2 (ODOSLToken *) - input
A reference to the second object in the comparison.

CallCompareProc Parameter - result

result (ODBoolean *) - output
A flag whose return value is kODTrue or kODFalse based the operator used for comparison and the objects it is comparing.

CallCompareProc - Return Value

None.

CallCompareProc - Parameters

thePart (ODPart *) - input
A reference to the part associated with this semantic-interface object.

oper (ODDescType) - input
The comparison operator that specifies how to compare the two objects.

obj1 (ODOSLToken *) - input
A reference to the first object in the comparison.

obj2 (ODOSLToken *) - input
A reference to the second object in the comparison.

result (ODBoolean *) - output

A flag whose return value is `KODTrue` or `KODFalse` based the operator used for comparison and the objects it is comparing.

None.

CallCompareProc - Remarks

OpenDoc calls this method during object resolution if an object specifier requires comparisons between a series of objects. Your override of this method is responsible for determining what comparisons make sense among your objects and should also be capable of comparing any two objects regardless of class type. For a list of the standard comparison operators that your method should be able to handle, see the chapter on resolving and creating object-specifier records in the *Open Scripting Architecture Guide and Reference for OS/2*.

You can use the name resolver's [IsODToken](#) method to determine whether the *obj1* and *obj2* parameters are OpenDoc tokens or just simple descriptors.

CallCompareProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to compare the specified objects.

This method can throw platform-specific exceptions.

CallCompareProc - Related Methods

Related Methods

- [ODNameResolver::IsODToken](#)

CallCompareProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CallCountProc

CallCountProc - Syntax

This method counts the number of elements of the specified type in the specified container.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *thePart;
ODDescType  desiredType;
ODDescType  containerClass;
ODOSLToken  *container;
ODSLong     *result;

CallCountProc(thePart, desiredType, containerClass,
              container, result);
```

CallCountProc Parameter - thePart

thePart (ODPart *) - input
A reference to a part associated with this semantic-interface object.

CallCountProc Parameter - desiredType

desiredType (ODDescType) - input
The type of object to be counted.

CallCountProc Parameter - containerClass

containerClass (ODDescType) - input
The object class of the container for the desired objects.

CallCountProc Parameter - container

container (ODOSLToken *) - input
A reference to an OpenDoc token for the container.

CallCountProc Parameter - result

result (ODSLong *) - output
The number of objects of the desired type in the container.

CallCountProc - Return Value

None.

CallCountProc - Parameters

thePart (ODPart *) - input
A reference to a part associated with this semantic-interface object.

desiredType (ODDescType) - input
The type of object to be counted.

containerClass (ODDescType) - input
The object class of the container for the desired objects.

container (ODOSLToken *) - input
A reference to an OpenDoc token for the container.

result (ODSLong *) - output
The number of objects of the desired type in the container.

None.

CallCountProc - Remarks

This method counts the elements in the container whose type matches the *desiredType* parameter and returns that number in the *result* parameter.

CallCountProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to count the specified type of object.

This method can throw platform-specific exceptions.

CallCountProc - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

CallDisposeTokenProc

CallDisposeTokenProc - Syntax

This method deallocates any part-specific data structures stored in the specified token.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart          *thePart;
ODOSLToken      *unneededToken;

CallDisposeTokenProc(thePart, unneededToken);
```

CallDisposeTokenProc Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

CallDisposeTokenProc Parameter - unneededToken

unneededToken (ODOSLToken *) - input

A reference to the OpenDoc token to be deleted.

CallDisposeTokenProc - Return Value

None.

CallDisposeTokenProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

unnneededToken (ODOSLToken *) - input

A reference to the OpenDoc token to be deleted.

None.

CallDisposeTokenProc - Remarks

You should override this method if your token contains any part-specific data structures. Your semantic interface should also supply this handler if it implements a set of marking methods. When one of your mark tokens is passed to this method, your override of this method should unmark the objects associated with that token and deallocate any necessary marking data structures.

CallDisposeTokenProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to deallocate the specifiec token.

This method can throw platform-specific exceptions.

CallDisposeTokenProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

CallEventHandler

CallEventHandler - Syntax

This method processes the specified OSA event object for the part.


```
#define INCL_ODSEMANTEICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *thePart;
ODOSAEvent  *theODOSAEvent;
ODOSAEvent  *reply;

CallEventHandler(thePart, theODOSAEvent, reply);
```

CallEventHandler Parameter - thePart

thePart (ODPart *) - input
A reference to the part associated with this semantic-interface object.

CallEventHandler Parameter - theODOSAEvent

theODOSAEvent (ODOSAEvent *) - input
A reference to the OSA event object to be processed.

CallEventHandler Parameter - reply

reply (ODOSAEvent *) - input
A reference to a reply OSA event object returned by the message-interface object's [Send](#) method.

CallEventHandler - Return Value

None.

CallEventHandler - Parameters

thePart (ODPart *) - input
A reference to the part associated with this semantic-interface object.

theODOSAEvent (ODOSAEvent *) - input
A reference to the OSA event object to be processed.

reply (ODOSAEvent *) - input

A reference to a reply OSA event object returned by the message-interface object's [Send](#) method.

None.

CallEventHandler - Remarks

This method processes the event, resolving any object specifiers along the way, and returns an appropriate reply in the provided OSA event object. This method should be capable of handling all the OSA event objects your part supports, responding to all events your part supports, and generating an appropriate exception if the specified OSA event object is not supported.

If any of the fields specified in the *theODOSAEvent* parameter are object specifiers, your method should resolve them using the name resolver's [Resolve](#) method.

CallEventHandler - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to handle the specified OSA event object.

This method can throw platform-specific exceptions.

CallEventHandler - Related Methods

Related Methods

- [ODNameResolver::Resolve](#)
 - [ODMessageInterface::Send](#)
-

CallEventHandler - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CallGetErrDescProc

CallGetErrDescProc - Syntax

This method returns a reference the part's global error descriptor object.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *thePart;
ODDesc      **errDesc;

CallGetErrDescProc(thePart, errDesc);
```

CallGetErrDescProc Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic interface object.

CallGetErrDescProc Parameter - errDesc

errDesc (ODDesc **) - output

A reference to the part's global error descriptor object.

CallGetErrDescProc - Return Value

None.

CallGetErrDescProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic interface object.

errDesc (ODDesc **) - output

A reference to the part's global error descriptor object.

None.

CallGetErrDescProc - Remarks

OpenDoc calls this method before an attempt is made to resolve an object specifier and again during object resolution if an exception arises. The first call of this method clears out the *errDesc* parameter and sets the descriptor type to typeNull. If an exception occurs during object resolution, the second call to this method fills in the *errDesc* parameter with the descriptor record that caused the exception.

Your override of this method should define the descriptor record as a global variable and use the same descriptor for all your exception handling. When your part receives an exception during object resolution, it can use your global descriptor record to get information about the exception.

CallGetErrDescProc - Exception Handling

The OSA Event Manager may throw an exception if this semantic interface does not support this functionality.

This method can throw platform-specific exceptions.

CallGetErrDescProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

CallGetMarkTokenProc

CallGetMarkTokenProc - Syntax

This method returns the mark token to be used for marking a large series of objects.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODPart          *thePart;
ODOSLToken      *dContainerToken;
ODDescType      containerClass;
ODOSLToken      *result;
```

```
CallGetMarkTokenProc(thePart, dContainerToken,
    containerClass, result);
```

CallGetMarkTokenProc Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

CallGetMarkTokenProc Parameter - dContainerToken

dContainerToken (ODOSLToken *) - input

A reference to the OpenDoc token that specifies the container of elements to be marked.

CallGetMarkTokenProc Parameter - containerClass

containerClass (ODDescType) - input

The object class of the container for the desired objects.

CallGetMarkTokenProc Parameter - result

result (ODOSLToken *) - output

A reference to the OpenDoc token to be used for marking the elements.

CallGetMarkTokenProc - Return Value

None.

CallGetMarkTokenProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

dContainerToken (ODOSLToken *) - input

A reference to the OpenDoc token that specifies the container of elements to be marked.

containerClass (ODDescType) - input

The object class of the container for the desired objects.

result (ODOSLToken *) - output
A reference to the OpenDoc token to be used for marking the elements.

None.

CallGetMarkTokenProc - Remarks

OpenDoc calls this method to obtain the mark token to pass to your semantic interface's [CallMarkProc](#) and [CallAdjustMarksProc](#) methods. The mark token you provide should contain information to uniquely identify the series of marked objects to your part.

Before OpenDoc can call this method, you must call your semantic-interface object's [SetOSLSupportFlags](#) method and specify the flag `kAEIDoMarking` to indicate that your semantic interface supports marking. In general, your part should override the [CallGetMarkTokenProc](#), [CallMarkProc](#), and [CallAdjustMarksProc](#) methods in the following cases: it already supports the marking of objects or it expects to deal with large numbers of records that might not all fit into memory at once.

CallGetMarkTokenProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to return an appropriate mark token.

This method may throw platform-specific exceptions.

CallGetMarkTokenProc - Related Methods

Related Methods

- [ODSemanticInterface::CallAdjustMarksProc](#)
- [ODSemanticInterface::CallMarkProc](#)
- [ODSemanticInterface::SetOSLSupportFlags](#)

CallGetMarkTokenProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CallMarkProc

CallMarkProc - Syntax

This method marks a large series of objects using the specified mark token.

```
#define INCL_ODSEMANTEICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *thePart;
ODOSLToken  *dToken;
ODOSLToken  *markToken;
ODSLong     index;

CallMarkProc(thePart, dToken, markToken, index);
```

CallMarkProc Parameter - thePart

thePart (ODPart *) - input
A reference to the part associated with this semantic-interface object.

CallMarkProc Parameter - dToken

dToken (ODOSLToken *) - input
A reference to the OpenDoc token that identifies the object to be marked.

CallMarkProc Parameter - markToken

markToken (ODOSLToken *) - in/out
A reference to an OpenDoc token to be used for marking the elements.

CallMarkProc Parameter - index

index (ODSLong) - input
The current mark count. The method should associate this index with each marked element.

CallMarkProc - Return Value

None.

CallMarkProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

dToken (ODOSLToken *) - input

A reference to the OpenDoc token that identifies the object to be marked.

markToken (ODOSLToken *) - in/out

A reference to an OpenDoc token to be used for marking the elements.

index (ODSLong) - input

The current mark count. The method should associate this index with each marked element.

None.

CallMarkProc - Remarks

OpenDoc calls this method once for each object that must be marked. Your override of this method should mark the object specified by the *dToken* parameter using the specified mark token and index. OpenDoc uses the index values to identify a range of objects to unmark when it calls the [CallAdjustMarksProc](#) method.

Before OpenDoc can call this method, you must call your semantic-interface object's [SetOSLSupportFlags](#) method and specify the flag `kAEIDoMarking` to indicate that your semantic interface supports marking. In general, your part should override the [CallGetMarkTokenProc](#), [CallMarkProc](#), and [CallAdjustMarksProc](#) methods in the following cases: it already supports the marking of objects or it expects to deal with large numbers of records that might not all fit into memory at once.

CallMarkProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to mark the specified object.

This method can throw platform-specific exceptions.

CallMarkProc - Related Methods

Related Methods

- [ODSemanticInterface::CallAdjustMarksProc](#)
 - [ODSemanticInterface::CallGetMarkTokenProc](#)
 - [ODSemanticInterface::SetOSLSupportFlags](#)
-

CallMarkProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

CallObjectAccessor

CallObjectAccessor - Syntax

This method resolves the object specifier into a target and returns a reference to an OpenDoc token identifying that target.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart          *thePart;
ODDescType      desiredClass;
ODOSLToken      *container;
ODDescType      containerClass;
ODDescType      form;
ODDesc          *selectionData;
ODOSLToken      *value;

CallObjectAccessor(thePart, desiredClass,
                  container, containerClass, form, selectionData,
                  value);
```

CallObjectAccessor Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

CallObjectAccessor Parameter - desiredClass

desiredClass (ODDescType) - input

The class of the desired OSA event objects.

CallObjectAccessor Parameter - container

container (ODOSLToken *) - input
A reference to an OpenDoc token identifying the container for the desired objects.

CallObjectAccessor Parameter - containerClass

containerClass (ODDescType) - input
The class of the container for the desired OSA event objects.

CallObjectAccessor Parameter - form

form (ODDescType) - input
The key form specified by the object specifier record for the object or objects to be located.

CallObjectAccessor Parameter - selectionData

selectionData (ODDesc *) - input
A reference to a descriptor object with the key data specified by the object specifier record for the object or objects to be located.

CallObjectAccessor Parameter - value

value (ODOSLToken *) - input
A reference to an OpenDoc token to be filled in by the object accessor being called.

CallObjectAccessor - Return Value

None.

CallObjectAccessor - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

desiredClass (ODDescType) - input

The class of the desired OSA event objects.

container (ODOSLToken *) - input

A reference to an OpenDoc token identifying the container for the desired objects.

containerClass (ODDescType) - input

The class of the container for the desired OSA event objects.

form (ODDescType) - input

The key form specified by the object specifier record for the object or objects to be located.

selectionData (ODDesc *) - input

A reference to a descriptor object with the key data specified by the object specifier record for the object or objects to be located.

value (ODOSLToken *) - input

A reference to an OpenDoc token to be filled in by the object accessor being called.

None.

CallObjectAccessor - Remarks

OpenDoc calls this method in response to a call to the name resolver's [Resolve](#) method. Your override of this method should be able to resolve any of the object types supported by your part.

Use the *desiredClass* and *containerClass* parameters to identify the appropriate handler.

If your part cannot resolve an object specifier, but one of your embedded parts might be able to, your override of this method can set the *value* parameter to be equivalent to the swap token obtained from the name resolver's [CreateSwapToken](#) method. This is important for parts that support embedding because your part's semantic interface must allow embedded parts to resolve objects that they know about.

For more information on resolving object specifiers, see the chapter on resolving and creating object specifier records in the *Open Scripting Architecture Guide and Reference for OS/2* .

CallObjectAccessor - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to resolve the specified object.

This method can throw platform-specific exceptions.

CallObjectAccessor - Related Methods

Related Methods

- [ODNameResolver::CreateSwapToken](#)
- [ODNameResolver::Resolve](#)

CallObjectAccessor - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

CallPredispatchProc

CallPredispatchProc - Syntax

This method calls the predispatch handler for this semantic interface's part.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart          *thePart;
ODOSAEvent      *theODOSAEvent;
ODOSAEvent      *reply;

CallPredispatchProc(thePart, theODOSAEvent,
                    reply);
```

CallPredispatchProc Parameter - thePart

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

CallPredispatchProc Parameter - theODOSAEvent

theODOSAEvent (ODOSAEvent *) - input

A reference to an OSA event object.

CallPredispatchProc Parameter - reply

reply (ODOSAEvent *) - input

A reference to an OSA event object reply that it is appropriate for the part to return.

CallPredispatchProc - Return Value

None.

CallPredispatchProc - Parameters

thePart (ODPart *) - input

A reference to the part associated with this semantic-interface object.

theODOSAEvent (ODOSAEvent *) - input

A reference to an OSA event object.

reply (ODOSAEvent *) - input

A reference to an OSA event object reply that it is appropriate for the part to return.

None.

CallPredispatchProc - Remarks

OpenDoc calls this method for each registered semantic interface to receive predispatched OSA event objects. This method gives your semantic interface a chance to react to events that can not be destined for your part. For example, OpenDoc does not forward recording-on and recording-off events to parts that are not direct recipients, so your part can use this method to detect those events.

Before OpenDoc calls your override of this method, you must call your semantic-interface object's [UsingPredispatchProc](#) method and pass the value kODTrue for the *usingNotUsing* parameter. Your override of this method should not raise any exceptions under normal conditions.

CallPredispatchProc - Exception Handling

The OSA Event Manager may throw an exception if this method is unable to receive predispatched OSA event objects.

This method can throw platform-specific exceptions.

CallPredispatchProc - Related Methods

Related Methods

- [ODSemanticInterface::UsingPredispatchProc](#)

CallPredispatchProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetOSLSupportFlags

GetOSLSupportFlags - Syntax

This method indicates which handlers this semantic-interface object supports.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODSShort    rv;
```

```
rv = GetOSLSupportFlags();
```

GetOSLSupportFlags Return Value - rv

rv ([ODSShort](#)) - returns

The flags representing the level of Object Support Library (OSL) support.

GetOSLSupportFlags - Parameters

rv ([ODSShort](#)) - returns

The flags representing the level of Object Support Library (OSL) support.

GetOSLSupportFlags - Remarks

This method returns the callback flags used when a part tries to resolve an object specifier using this semantic-interface object.

GetOSLSupportFlags - Related Methods

Related Methods

- [ODNameResolver::Resolve](#)
 - [ODSemanticInterface::SetOSLSupportFlags](#)
-

GetOSLSupportFlags - Topics

Class:

[ODSemanticInterface](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

InitBaseSemanticInterface

InitBaseSemanticInterface - Syntax

This method initializes this object.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODPart      *base;
ODSession   *session;
```

```
InitBaseSemanticInterface(base, session);
```

InitBaseSemanticInterface Parameter - base

base (ODPart *) - input
The part to which this semantic-interface object belongs.

InitBaseSemanticInterface Parameter - session

session (ODSession *) - input
A reference to the current session object.

InitBaseSemanticInterface - Return Value

None.

InitBaseSemanticInterface - Parameters

base (ODPart *) - input
The part to which this semantic-interface object belongs.

session (ODSession *) - input
A reference to the current session object.

None.

InitBaseSemanticInterface - Remarks

This method is called by [InitSemanticInterface](#) and should not be called directly.

InitBaseSemanticInterface - Exception Handling

kODErrOutOfMemory

There is not enough memory to initialize this object.

InitBaseSemanticInterface - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

InitSemanticInterface

InitSemanticInterface - Syntax

This method initializes this semantic-interface object.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODPart      *base;
ODSession   *session;

InitSemanticInterface(base, session);
```

InitSemanticInterface Parameter - base

base (ODPart *) - input
A reference to a part associated with this semantic-interface object.

InitSemanticInterface Parameter - session

session (ODSession *) - input
A reference to the current session object.

InitSemanticInterface - Return Value

None.

InitSemanticInterface - Parameters

base (ODPart *) - input

A reference to a part associated with this semantic-interface object.

session (ODSession *) - input

A reference to the current session object.

None.

InitSemanticInterface - Remarks

This method is not called directly to initialize this semantic interface object, but is called by a subclass-specific initialization method. By convention, every subclass of [ODSemanticInterface](#) should have a separate initialization method (for example, `InitMySemanticInterface` method) that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the `InitSemanticInterface` method. The `InitMySemanticInterface` method should call the inherited `InitSemanticInterface` method at the beginning of its implementation.

If you subclass [ODSemanticInterface](#), your subclass-specific initialization method, rather than its `somInit` method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your semantic interface.

InitSemanticInterface - Override Policy

If you subclass [ODSemanticInterface](#), you must not override this method.

InitSemanticInterface - Topics

Class:

[ODSemanticInterface](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

InstallAdjustMarksProc

InstallAdjustMarksProc - Syntax

This method installs an AdjustMarksProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODAdjustMarksUPP    adjustMarksProc;
ODSLong             refCon;

InstallAdjustMarksProc (adjustMarksProc, refCon);
```

InstallAdjustMarksProc Parameter - adjustMarksProc

adjustMarksProc (ODAdjustMarksUPP) - input
A reference to an AdjustMarksProc function.

InstallAdjustMarksProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallAdjustMarksProc - Return Value

None.

InstallAdjustMarksProc - Parameters

adjustMarksProc (ODAdjustMarksUPP) - input
A reference to an AdjustMarksProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallAdjustMarksProc - Topics

Class:
ODSemanticInterface

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

InstallCoercionHandler

InstallCoercionHandler - Syntax

This method installs a coercion handler.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODDescType      fromType;
ODDescType      toType;
ODCoercionHandlerUPP handler;
ODSLong         handlerRefCon;
ODBoolean       fromTypeIsDesc;

InstallCoercionHandler(fromType, toType, handler,
                      handlerRefCon, fromTypeIsDesc);
```

InstallCoercionHandler Parameter - fromType

fromType ([ODDescType](#)) - input
The type of descriptor to be coerced.

InstallCoercionHandler Parameter - toType

toType ([ODDescType](#)) - input
The type of descriptor to be created.

InstallCoercionHandler Parameter - handler

handler (ODCoercionHandlerUPP) - input
A reference to the coercion handler routine.

InstallCoercionHandler Parameter - handlerRefCon

handlerRefCon (ODSLong) - input
A user-defined reference constant.

InstallCoercionHandler Parameter - fromTypelsDesc

fromTypelsDesc (ODBoolean) - input
A flag indicating whether the data to be coerced is a descriptor.

kODTrue	The data to be coerced is a descriptor.
kODFalse	The data to be coerced is not a descriptor.

InstallCoercionHandler - Return Value

None.

InstallCoercionHandler - Parameters

fromType (ODDescType) - input
The type of descriptor to be coerced.

toType (ODDescType) - input
The type of descriptor to be created.

handler (ODCoercionHandlerUPP) - input
A reference to the coercion handler routine.

handlerRefCon (ODSLong) - input
A user-defined reference constant.

fromTypelsDesc (ODBoolean) - input
A flag indicating whether the data to be coerced is a descriptor.

kODTrue	The data to be coerced is a descriptor.
kODFalse	The data to be coerced is not a descriptor.

None.

InstallCoercionHandler - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InstallCompareProc

InstallCompareProc - Syntax

This method installs a CompareProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODCompareUPP    compareProc;
ODSLong         refCon;

InstallCompareProc(compareProc, refCon);
```

InstallCompareProc Parameter - compareProc

compareProc (ODCompareUPP) - input
A reference to the CompareProc function.

InstallCompareProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallCompareProc - Return Value

None.

InstallCompareProc - Parameters

compareProc (ODCompareUPP) - input
A reference to the CompareProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallCompareProc - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

InstallCountProc

InstallCountProc - Syntax

This method installs a CountProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODCountUPP    countProc;
ODSLong       refCon;

InstallCountProc(countProc, refCon);
```

InstallCountProc Parameter - countProc

countProc (ODCountUPP) - input
A reference to a CountProc function.

InstallCountProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallCountProc - Return Value

None.

InstallCountProc - Parameters

countProc (ODCountUPP) - input
A reference to a CountProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallCountProc - Topics

Class:
ODSemanticInterface

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

InstallDisposeTokenProc

InstallDisposeTokenProc - Syntax

This method installs a DisposeTokenProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODDisposeTokenUPP    disposeTokenProc;
ODSLong              refCon;

InstallDisposeTokenProc (disposeTokenProc,
                        refCon);
```

InstallDisposeTokenProc Parameter - disposeTokenProc

disposeTokenProc (ODDisposeTokenUPP) - input
A reference to a DisposeTokenProc function.

InstallDisposeTokenProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallDisposeTokenProc - Return Value

None.

InstallDisposeTokenProc - Parameters

disposeTokenProc (ODDisposeTokenUPP) - input
A reference to a DisposeTokenProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallDisposeTokenProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InstallEventHandler

InstallEventHandler - Syntax

This method installs an event handler.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

OEventClass      theAEEEventClass;
OEventID         theAEEEventID;
OEventHandlerUPP  handler;
ODSLong          handlerRefCon;

InstallEventHandler(theAEEEventClass, theAEEEventID,
                  handler, handlerRefCon);
```

InstallEventHandler Parameter - theAEEEventClass

theAEEEventClass ([OEventClass](#)) - input

The class of the event this handler is to support; for example, kCoreEventClass.

InstallEventHandler Parameter - theAEEEventID

theAEEEventID ([OEventID](#)) - input

The event ID this handler is to support; for example, kAEPrintDocument.

InstallEventHandler Parameter - handler

handler (ODEventHandlerUPP) - input
A reference to an event handler.

InstallEventHandler Parameter - handlerRefCon

handlerRefCon (ODSLong) - input
A user-defined reference constant.

InstallEventHandler - Return Value

None.

InstallEventHandler - Parameters

theAEEEventClass (ODEventClass) - input
The class of the event this handler is to support; for example, kCoreEventClass.

theAEEEventID (ODEventID) - input
The event ID this handler is to support; for example, kAEPrintDocument.

handler (ODEventHandlerUPP) - input
A reference to an event handler.

handlerRefCon (ODSLong) - input
A user-defined reference constant.

None.

InstallEventHandler - Topics

Class:
ODSemanticInterface

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

InstallGetErrDescProc

InstallGetErrDescProc - Syntax

This method installs a GetErrDescProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODGetErrDescUPP    errorDescProc;
ODSLong            refCon;

InstallGetErrDescProc(errorDescProc, refCon);
```

InstallGetErrDescProc Parameter - errorDescProc

errorDescProc (ODGetErrDescUPP) - input
A reference to a GetErrDescProc function.

InstallGetErrDescProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallGetErrDescProc - Return Value

None.

InstallGetErrDescProc - Parameters

errorDescProc (ODGetErrDescUPP) - input
A reference to a GetErrDescProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallGetErrDescProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InstallGetMarkTokenProc

InstallGetMarkTokenProc - Syntax

The method installs the GetMarkTokenProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODGetMarkTokenUPP    getMarkTokenProc;
ODSLong              refCon;

InstallGetMarkTokenProc (getMarkTokenProc,
                        refCon);
```

InstallGetMarkTokenProc Parameter - getMarkTokenProc

getMarkTokenProc (ODGetMarkTokenUPP) - input
A reference to the GetMarkTokenProc function.

InstallGetMarkTokenProc Parameter - refCon

refCon ([ODSLong](#)) - input
A user-defined reference constant.

InstallGetMarkTokenProc - Return Value

None.

InstallGetMarkTokenProc - Parameters

getMarkTokenProc (ODGetMarkTokenUPP) - input
A reference to the GetMarkTokenProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallGetMarkTokenProc - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

InstallMarkProc

InstallMarkProc - Syntax

This method installs a MarkProc object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODMarkUPP    markProc;
ODSLong      refCon;

InstallMarkProc(markProc, refCon);
```

InstallMarkProc Parameter - markProc

markProc (ODMarkUPP) - input
A reference to a MarkProc function.

InstallMarkProc Parameter - refCon

refCon (ODSLong) - input
A user-defined reference constant.

InstallMarkProc - Return Value

None.

InstallMarkProc - Parameters

markProc (ODMarkUPP) - input
A reference to a MarkProc function.

refCon (ODSLong) - input
A user-defined reference constant.

None.

InstallMarkProc - Topics

Class:
ODSemanticInterface

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

InstallObjectAccessor

InstallObjectAccessor - Syntax

This method installs an object accessor function.

```
#define INCL_ODSEMANANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODDescType      desiredClass;
ODDescType      containerType;
ODObjectAccessorUPP theAccessor;
ODSLong         accessorRefcon;

InstallObjectAccessor(desiredClass, containerType,
                     theAccessor, accessorRefcon);
```

InstallObjectAccessor Parameter - desiredClass

desiredClass (ODDescType) - input
The object class of the desired OSA event object.

InstallObjectAccessor Parameter - containerType

containerType (ODDescType) - input
The object class of the container for the desired class.

InstallObjectAccessor Parameter - theAccessor

theAccessor (ODObjectAccessorUPP) - input
A reference to the object accessor function.

InstallObjectAccessor Parameter - accessorRefcon

accessorRefcon (ODSLong) - input
A user-defined reference constant.

InstallObjectAccessor - Return Value

None.

InstallObjectAccessor - Parameters

desiredClass ([ODDescType](#)) - input

The object class of the desired OSA event object.

containerType ([ODDescType](#)) - input

The object class of the container for the desired class.

theAccessor ([ODObjAccessorUPP](#)) - input

A reference to the object accessor function.

accessorRefcon ([ODSLong](#)) - input

A user-defined reference constant.

None.

InstallObjectAccessor - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InstallSpecialHandler

InstallSpecialHandler - Syntax

This method installs an object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
AEKeyword          functionClass;
ODSpecialHandlerUPP handler;
ODSLong            refCon;
```

```
InstallSpecialHandler(functionClass, handler,
                      refCon);
```

InstallSpecialHandler Parameter - functionClass

functionClass ([AEKeyword](#)) - input

A constant defining the type of callback function to be installed. This parameter can be set to one of the following values:

keyAECountProc	The CountProc callback function.
keyAECompareProc	The CompareProc callback function.
keyAEDisposeProc	The DisposeProc callback function.
keyAEGetErrDescProc	The GetErrDescProc callback function.
keyAEMarkTokenProc	The MarkTokenProc callback function.
keyAEMarkProc	The MarkProc callback function.
keyAEAdjustMarksProc	The AdjustMarksProc callback function.
keyPreDispatch	The PreDispatch function.

InstallSpecialHandler Parameter - handler

handler ([ODSpecialHandlerUPP](#)) - input

A reference to an object callback function.

InstallSpecialHandler Parameter - refCon

refCon ([ODSLong](#)) - input

A user-defined reference constant.

InstallSpecialHandler - Return Value

None.

InstallSpecialHandler - Parameters

functionClass ([AEKeyword](#)) - input

A constant defining the type of callback function to be installed. This parameter can be set to one of the following values:

keyAECCountProc The CountProc callback function.
keyAECCompareProc The CompareProc callback function.
keyAEDisposeProc The DisposeProc callback function.
keyAEGetErrDescProc The GetErrDescProc callback function.
keyAEMarkTokenProc The MarkTokenProc callback function.
keyAEMarkProc The MarkProc callback function.
keyAEAdjustMarksProc The AdjustMarksProc callback function.
keyPreDispatch The PreDispatch function.

handler (ODSpecialHandlerUPP) - input

A reference to an object callback function.

refCon ([ODSLong](#)) - input

A user-defined reference constant.

None.

InstallSpecialHandler - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

RemoveCoercionHandler

RemoveCoercionHandler - Syntax

This method removes the specified coercion handler.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODDescType               fromType;
ODDescType               toType;
ODCoercionHandlerUPP     handler;
```

```
RemoveCoercionHandler(fromType, toType, handler);
```

RemoveCoercionHandler Parameter - fromType

fromType ([ODDescType](#)) - input
The type of descriptor to be coerced.

RemoveCoercionHandler Parameter - toType

toType ([ODDescType](#)) - input
The type of descriptor to be created.

RemoveCoercionHandler Parameter - handler

handler (ODCoercionHandlerUPP) - input
A reference to the coercion handler routine.

RemoveCoercionHandler - Return Value

None.

RemoveCoercionHandler - Parameters

fromType ([ODDescType](#)) - input
The type of descriptor to be coerced.

toType ([ODDescType](#)) - input
The type of descriptor to be created.

handler (ODCoercionHandlerUPP) - input
A reference to the coercion handler routine.

None.

RemoveCoercionHandler - Topics

Class:
ODSemanticInterface

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

RemoveEventHandler

RemoveEventHandler - Syntax

This method removes a specified event handler.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODEventClass      theAEEEventClass;
ODEventID         theAEEEventID;
ODEventHandlerUPP handler;

RemoveEventHandler(theAEEEventClass, theAEEEventID,
                  handler);
```

RemoveEventHandler Parameter - theAEEEventClass

theAEEEventClass ([ODEventClass](#)) - input
The class of the event supported by this handler.

RemoveEventHandler Parameter - theAEEEventID

theAEEEventID ([ODEventID](#)) - input
The event ID supported by this handler.

RemoveEventHandler Parameter - handler

handler (ODEventHandlerUPP) - input
A reference to the event handler.

RemoveEventHandler - Return Value

None.

RemoveEventHandler - Parameters

theAEEEventClass ([ODEventClass](#)) - input
The class of the event supported by this handler.

theAEEEventID ([ODEventID](#)) - input
The event ID supported by this handler.

handler (ODEventHandlerUPP) - input
A reference to the event handler.

None.

RemoveEventHandler - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

RemoveObjectAccessor

RemoveObjectAccessor - Syntax

This method removes an object accessor function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODDescType          desiredClass;
ODDescType          containerType;
```

```
ODObjectAccessorUPP theAccessor;  
  
RemoveObjectAccessor(desiredClass, containerType,  
    theAccessor);
```

RemoveObjectAccessor Parameter - desiredClass

desiredClass (ODDescType) - input
The object class of the desired OSA event object.

RemoveObjectAccessor Parameter - containerType

containerType (ODDescType) - input
The object class of the container for the desired class.

RemoveObjectAccessor Parameter - theAccessor

theAccessor (ODObjectAccessorUPP) - input
A reference to the object accessor function.

RemoveObjectAccessor - Return Value

None.

RemoveObjectAccessor - Parameters

desiredClass (ODDescType) - input
The object class of the desired OSA event object.

containerType (ODDescType) - input
The object class of the container for the desired class.

theAccessor (ODObjectAccessorUPP) - input
A reference to the object accessor function.

None.

RemoveObjectAccessor - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

RemoveSpecialHandler

RemoveSpecialHandler - Syntax

This method removes an object callback function.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

AEKeyword          functionClass;
ODSpecialHandlerUPP handler;

RemoveSpecialHandler(functionClass, handler);
```

RemoveSpecialHandler Parameter - functionClass

functionClass ([AEKeyword](#)) - input

A constant defining the type of callback function to be removed. This parameter can be set to one of the following values:

keyAECCountProc	The CountProc callback function.
keyAECompareProc	The CompareProc callback function.
keyAEDisposeProc	The DisposeProc callback function.
keyAEGetErrDescProc	The GetErrDescProc callback function.
keyAEMarkTokenProc	The MarkTokenProc callback function.
keyAEMarkProc	The MarkProc callback function.
keyAEAdjustMarksProc	The AdjustMarksProc callback function.
keyPreDispatch	The PreDispatch function.

RemoveSpecialHandler Parameter - handler

handler (ODSpecialHandlerUPP) - input
A reference to an object callback function.

RemoveSpecialHandler - Return Value

None.

RemoveSpecialHandler - Parameters

functionClass ([AEKeyword](#)) - input
A constant defining the type of callback function to be removed. This parameter can be set to one of the following values:

keyAECCountProc	The CountProc callback function.
keyAECompareProc	The CompareProc callback function.
keyAEDisposeProc	The DisposeProc callback function.
keyAEGetErrDescProc	The GetErrDescProc callback function.
keyAEMarkTokenProc	The MarkTokenProc callback function.
keyAEMarkProc	The MarkProc callback function.
keyAEAdjustMarksProc	The AdjustMarksProc callback function.
keyPreDispatch	The PreDispatch function.

handler (ODSpecialHandlerUPP) - input
A reference to an object callback function.

None.

RemoveSpecialHandler - Topics

Class:
ODSemanticInterface

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

SetOSLSupportFlags

SetOSLSupportFlags - Syntax

This method sets the flags that indicate which handlers this semantic-interface object supports.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODSShort    flags;

SetOSLSupportFlags(flags);
```

SetOSLSupportFlags Parameter - flags

flags ([ODSShort](#)) - input
The flags representing the level of Object Support Library (OSL) support.

SetOSLSupportFlags - Return Value

None.

SetOSLSupportFlags - Parameters

flags ([ODSShort](#)) - input
The flags representing the level of Object Support Library (OSL) support.

None.

SetOSLSupportFlags - Remarks

Your part calls this method to specify the callback flags to be used during the resolution of an object specifier.

For more information on resolving object specifiers and the use of callback flags, see the chapter on resolving and creating object specifier records in the *Open Scripting Architecture Guide and Reference for OS/2* .

SetOSLSupportFlags - Related Methods

Related Methods

- [ODNameResolver::Resolve](#)
- [ODSemanticInterface::GetOSLSupportFlags](#)

SetOSLSupportFlags - Topics

Class:

[ODSemanticInterface](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

UsingPredispatchProc

UsingPredispatchProc - Syntax

This method specifies whether the predispatch method is currently being called whenever OpenDoc receives an OSA event.

```
#define INCL_ODSEMANTICINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODBoolean    usingNotUsing;

UsingPredispatchProc(usingNotUsing);
```

UsingPredispatchProc Parameter - usingNotUsing

usingNotUsing ([ODBoolean](#)) - input

A flag indication whether the predispatch method is currently being called whenever OpenDoc receives an OSA event.

kODTrue

The predispatch method is currently being called whenever OpenDoc receives an OSA event.

kODFalse

The predispatch method is not currently being called whenever OpenDoc receives an OSA event.

UsingPredispatchProc - Return Value

None.

UsingPredispatchProc - Parameters

usingNotUsing ([ODBoolean](#)) - input

A flag indication whether the predispatch method is currently being called whenever OpenDoc receives an OSA event.

kODTrue

The predispatch method is currently being called whenever OpenDoc receives an OSA event.

kODFalse

The predispatch method is not currently being called whenever OpenDoc receives an OSA event.

None.

UsingPredispatchProc - Remarks

You must call this method, passing the value of kODTrue to the *usingNotUsing* parameter, before OpenDoc can call your semantic interface's [CallPredispatchProc](#) method.

UsingPredispatchProc - Topics

Class:

ODSemanticInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

ODSession

Class Definition File: ODSESSN.IDL

Class Hierarchy

SOMObject
ODObject
ODBaseSession
ODSession

Description

The ODSession class provides access to session-wide OpenDoc objects as well as the initialization and shutdown of the OpenDoc environment.

When an OpenDoc document is opened, the document shell creates and initializes a single instance of ODSession. (A part editor should never create an instance of this class directly.) During its initialization, the session object creates a single object of several OpenDoc classes. These OpenDoc objects provide the environment for supporting and manipulating the document.

Through the session object, a part editor can obtain references to most of the OpenDoc objects: the arbitrator, the clipboard object, the dispatcher, the drag-and-drop object, the information object, the message interface, the name resolver, the name-space manager, the storage system, the translation object, the undo object, and the window-state object. The document shell can similarly obtain references to the binding object, the link manager, and the document shell's semantic interface.

The ODSession class also includes methods for converting between a type string (an [ODType](#)) and the corresponding token (an [ODTokenType](#)), removing an entry from the type/token table and accessing the name of the current user of the document. The session object also generates link update IDs, which parts use to prevent circular updating when they synchronize linked data.

For more information about the OpenDoc objects that part editors can access, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*. For more information about linking and link update IDs, see the chapters on storage and data transfer in the *OpenDoc Programming Guide*.

Methods

The methods defined by the ODSession class include:

Initializing [InitSession](#)

Accessing Global Objects

- [AcquireShellSemtInterface](#)
- [GetArbitrator](#)
- [GetBinding](#)
- [GetClipboard](#)
- [GetDispatcher](#)
- [GetDragAndDrop](#)
- [GetHelp](#)
- [GetInfo](#)
- [GetMessageInterface](#)
- [GetNameResolver](#)
- [GetNameSpaceManager](#)
- [GetStorageSystem](#)
- [GetTranslation](#)
- [GetToolSpaceManager](#)
- [GetUndo](#)
- [GetWindowState](#)

Replacing Global Objects

- [SetArbitrator](#)
- [SetBinding](#)
- [SetClipboard](#)
- [SetDispatcher](#)
- [SetDragAndDrop](#)
- [SetInfo](#)
- [SetLinkManager](#)
- [SetMessageInterface](#)
- [SetNameResolver](#)
- [SetNameSpaceManager](#)
- [SetShellSemtInterface](#)
- [SetStorageSystem](#)
- [SetToolSpaceManager](#)
- [SetTranslation](#)
- [SetUndo](#)
- [SetWindowState](#)

Converting Between Type Strings and Tokens

- [GetType](#)
- [RemoveEntry](#)
- [Tokenize](#)

Miscellaneous

- [CreatePlatformWindow](#)
- [GetUserName](#)
- [UniqueUpdateID](#)

Overridden Methods

There are currently no methods overridden by the ODSession class.

AcquireShellSemtInterface

AcquireShellSemtInterface - Syntax

This method is called by the document shell or container application to retrieve a reference to the document shell's semantic interface object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODSemanticInterface      *rv;

rv = AcquireShellSemtInterface();
```

AcquireShellSemtInterface Return Value - rv

rv (ODSemanticInterface *) - returns
A reference to the document shell's semantic interface object.

AcquireShellSemtInterface - Parameters

rv (ODSemanticInterface *) - returns
A reference to the document shell's semantic interface object.

AcquireShellSemtInterface - Remarks

This method increments the reference count of the returned semantic interface object. When you have finished using that semantic interface object, you should call its [Release](#) method.

For more information on semantic interfaces, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

AcquireShellSemtInterface - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

CreatePlatformWindow (OS/2)

CreatePlatformWindow (OS/2) - Syntax

This method is called to ask the document shell or container application to create a document window.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindowCreateOptions    options;
ODPlatformWindow                rv;

rv = CreatePlatformWindow(options);
```

CreatePlatformWindow (OS/2) Parameter - options

options ([ODPlatformWindowCreateOptions](#)) - input

This parameter can be set to the following value:

FCF_ACCELTABLE

An accelerator table is required.

FCF_SHELLPOSITION

The standard size, coordinates, and position are determined by the system.

FCF_STANDARD

A menu-template resource is required.

CreatePlatformWindow (OS/2) Return Value - rv

rv ([ODPlatformWindow](#)) - returns

The document window just created.

CreatePlatformWindow (OS/2) - Parameters

options ([ODPlatformWindowCreateOptions](#)) - input

This parameter can be set to the following value:

FCF_ACCELTABLE

An accelerator table is required.

FCF_SHELLPOSITION

The standard size, coordinates, and position are determined by the system.

FCF_STANDARD

A menu-template resource is required.

rv ([ODPlatformWindow](#)) - returns

The document window just created.

CreatePlatformWindow (OS/2) - Remarks

On OS/2, the window procedure for the document window is contained in the document shell or container applications. This means the OpenDoc parts and OpenDoc run-time cannot themselves create document windows because they do not know the address of the window procedure which must be specified as a parameter to the WinCreateWindow function.

The CreatePlatformWindow method is used by OpenDoc parts and the OpenDoc run-time to create document windows using platform APIs. The base class implementation of this method performs no action. It is intended to be overridden and implemented by the document shell or container applications when they instantiate a session object which inherits from the [ODSession](#) class.

CreatePlatformWindow (OS/2) - Override Policy

If you subclass [ODSession](#), you must override this method. This method must be implemented by the document shell or container application that creates the [ODSession](#) object.

CreatePlatformWindow (OS/2) - Topics

Class:

[ODSession](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

GetArbitrator

GetArbitrator - Syntax

This method returns a reference to the arbitrator for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODArbitrator      *rv;

rv = GetArbitrator();
```

GetArbitrator Return Value - rv

rv (ODArbitrator *) - returns
A reference to the arbitrator.

GetArbitrator - Parameters

rv (ODArbitrator *) - returns
A reference to the arbitrator.

GetArbitrator - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetBinding

GetBinding - Syntax

This method is called by the document shell or container application to retrieve a reference to the binding object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>

ODBinding      *rv;

rv = GetBinding();
```

GetBinding Return Value - rv

rv (ODBinding *) - returns
A reference to the binding object.

GetBinding - Parameters

rv (ODBinding *) - returns
A reference to the binding object.

GetBinding - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetClipboard

GetClipboard - Syntax

This method returns a reference to the clipboard object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODClipboard      *rv;  
  
rv = GetClipboard();
```

GetClipboard Return Value - rv

rv (ODClipboard *) - returns
A reference to the clipboard object.

GetClipboard - Parameters

rv (ODClipboard *) - returns
A reference to the clipboard object.

GetClipboard - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetDispatcher

GetDispatcher - Syntax

This method returns a reference to the dispatcher for this session.

```
#define INCL_ODSESSION  
#define INCL_ODAPI  
#include <os2.h>  
  
ODDispatcher      *rv;  
  
rv = GetDispatcher();
```

GetDispatcher Return Value - rv

rv (ODDispatcher *) - returns
A reference to the dispatcher.

GetDispatcher - Parameters

rv (ODDispatcher *) - returns
A reference to the dispatcher.

GetDispatcher - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetDragAndDrop

GetDragAndDrop - Syntax

This method returns a reference to the drag-and-drop object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODDragAndDrop *rv;

rv = GetDragAndDrop();
```

GetDragAndDrop Return Value - rv

rv (ODDragAndDrop *) - returns
A reference to the drag-and-drop object.

GetDragAndDrop - Parameters

rv (ODDragAndDrop *) - returns
A reference to the drag-and-drop object.

GetDragAndDrop - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetHelp (OS/2)

GetHelp (OS/2) - Syntax

This method returns a reference to the help object for this session.

```
#define INCL_ODSESSION  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODHelp      *rv;
```

```
rv = GetHelp();
```

GetHelp (OS/2) Return Value - rv

rv (ODHelp *) - returns
A reference to the help object.

GetHelp (OS/2) - Parameters

rv (ODHelp *) - returns
A reference to the help object.

GetHelp (OS/2) - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetInfo

GetInfo - Syntax

This method returns a reference to the information object for this session.

```
#define INCL_ODSESSION  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODInfo      *rv;
```

```
rv = GetInfo();
```

GetInfo Return Value - rv

rv (ODInfo *) - returns
A reference to the information object.

GetInfo - Parameters

rv (ODInfo *) - returns
A reference to the information object.

GetInfo - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetMessageInterface

GetMessageInterface - Syntax

This method returns a reference to the message interface for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODMessageInterface      *rv;

rv = GetMessageInterface();
```

GetMessageInterface Return Value - rv

rv (ODMessageInterface *) - returns
A reference to the message interface.

GetMessageInterface - Parameters

rv (ODMessageInterface *) - returns
A reference to the message interface.

GetMessageInterface - Topics

Class:

ODSession

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)

GetNameResolver

GetNameResolver - Syntax

This method returns a reference to the name resolver object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODNameResolver      *rv;

rv = GetNameResolver();
```

GetNameResolver Return Value - rv

rv (ODNameResolver *) - returns
A reference to the name resolver.

GetNameResolver - Parameters

rv (ODNameResolver *) - returns
A reference to the name resolver.

GetNameResolver - Topics

Class:

ODSession

Select an item:

[Syntax](#)

GetNameSpaceManager

GetNameSpaceManager - Syntax

This method returns a reference to the name-space manager for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODNameSpaceManager      *rv;

rv = GetNameSpaceManager();
```

GetNameSpaceManager Return Value - rv

rv (ODNameSpaceManager *) - returns
A reference to the name-space manager.

GetNameSpaceManager - Parameters

rv (ODNameSpaceManager *) - returns
A reference to the name-space manager.

GetNameSpaceManager - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetStorageSystem

GetStorageSystem - Syntax

This method returns a reference to the storage system for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODStorageSystem      *rv;

rv = GetStorageSystem();
```

GetStorageSystem Return Value - rv

rv (ODStorageSystem *) - returns
A reference to the storage system.

GetStorageSystem - Parameters

rv (ODStorageSystem *) - returns
A reference to the storage system.

GetStorageSystem - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetTranslation

GetTranslation - Syntax

This method returns a reference to the translation object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODTranslation      *rv;

rv = GetTranslation();
```

GetTranslation Return Value - rv

rv (ODTranslation *) - returns
A reference to the translation object.

GetTranslation - Parameters

rv (ODTranslation *) - returns
A reference to the translation object.

GetTranslation - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetToolSpaceManager (OS/2)

GetToolSpaceManager (OS/2) - Syntax

This method returns a reference to the tool-space manager for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
```

```
#include <os2.h>

ODToolSpaceManager    *rv;

rv = GetToolSpaceManager();
```

GetToolSpaceManager (OS/2) Return Value - rv

rv (ODToolSpaceManager *) - returns
A reference to the tool-space manager.

GetToolSpaceManager (OS/2) - Parameters

rv (ODToolSpaceManager *) - returns
A reference to the tool-space manager.

GetToolSpaceManager (OS/2) - Topics

Class:
ODSession

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetType

GetType - Syntax

This method gets the type string corresponding to the specified token, if the token exists.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    token;
ODType         *type;
ODBoolean      rv;

rv = GetType(token, type);
```

GetType Parameter - token

token (ODTypeToken) - input
A tokenized string representing the token of interest.

GetType Parameter - type

type (ODType *) - output
The type string corresponding to the specified token.

GetType Return Value - rv

rv (ODBoolean) - returns
A flag indicating whether the specified token exists in the type/token table for this session.

kODTrue	The specified token exists in the type/token table for this session.
kODFalse	The specified token does not exist in the type/token table for this session.

GetType - Parameters

token (ODTypeToken) - input
A tokenized string representing the token of interest.

type (ODType *) - output
The type string corresponding to the specified token.

rv (ODBoolean) - returns
A flag indicating whether the specified token exists in the type/token table for this session.

kODTrue	The specified token exists in the type/token table for this session.
kODFalse	The specified token does not exist in the type/token table for this session.

GetType - Related Methods

Related Methods

- [ODSession::RemoveEntry](#)
- [ODSession::Tokenize](#)

GetType - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

GetUndo

GetUndo - Syntax

This method returns a reference to the undo object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODUndo      *rv;
```

```
rv = GetUndo();
```

GetUndo Return Value - rv

rv (ODUndo *) - returns

A reference to the undo object.

GetUndo - Parameters

rv (ODUndo *) - returns

A reference to the undo object.

GetUndo - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetUserName

GetUserName - Syntax

This method returns a text string identifying the current user of the document.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODIText      *name;
```

```
GetUserName (name);
```

GetUserName Parameter - name

name ([ODIText](#) *) - output

A string identifying the current user of the document.

GetUserName - Return Value

None.

GetUserName - Parameters

name ([ODIText *](#)) - output
A string identifying the current user of the document.

None.

GetUserName - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetWindowState

GetWindowState - Syntax

This methods returns a reference to the window-state object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODWindowState      *rv;

rv = GetWindowState();
```

GetWindowState Return Value - rv

rv ([ODWindowState *](#)) - returns
A reference to the window-state object.

GetWindowState - Parameters

rv ([ODWindowState *](#)) - returns
A reference to the window-state object.

GetWindowState - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InitSession

InitSession - Syntax

This method initializes this session and creates its objects.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
InitSession();
```

InitSession - Return Value

None.

InitSession - Parameters

None.

InitSession - Remarks

Your part editor should never call this method directly; the document shell automatically calls this method when an OpenDoc document is opened.

InitSession - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

RemoveEntry

RemoveEntry - Syntax

This method removes the specified entry from the type/token table for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODType    type;
```

```
RemoveEntry(type);
```

RemoveEntry Parameter - type

type ([ODType](#)) - in/out

The type string identifying the entry to be removed from the type/token table.

RemoveEntry - Return Value

None.

RemoveEntry - Parameters

type ([ODType](#)) - in/out

The type string identifying the entry to be removed from the type/token table.

None.

RemoveEntry - Remarks

If the type string was never converted to a token, no action is taken (because the table does not contain an entry for the specified type).

RemoveEntry - Related Methods

Related Methods

- - [ODSession::GetType](#)
 - [ODSession::Tokenize](#)
-

RemoveEntry - Topics

Class:

[ODSession](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

SetArbitrator

SetArbitrator - Syntax

This method replaces the arbitrator object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODArbitrator      *arbitrator;

SetArbitrator(arbitrator);
```

SetArbitrator Parameter - arbitrator

arbitrator (ODArbitrator *) - input
A reference to the new arbitrator for this session.

SetArbitrator - Return Value

None.

SetArbitrator - Parameters

arbitrator (ODArbitrator *) - input
A reference to the new arbitrator for this session.

None.

SetArbitrator - Remarks

This method may be called only by a shell plug-in's installation function.

SetArbitrator - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetBinding

SetBinding - Syntax

This method replaces the binding object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>

ODBinding      *binding;

SetBinding(binding);
```

SetBinding Parameter - binding

binding (ODBinding *) - input
A reference to the new binding object for this session.

SetBinding - Return Value

None.

SetBinding - Parameters

binding (ODBinding *) - input
A reference to the new binding object for this session.

None.

SetBinding - Remarks

This method may be called only by a shell plug-in's installation function.

SetBinding - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetClipboard

SetClipboard - Syntax

This method replaces the clipboard object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>

ODClipboard      *clipboard;

SetClipboard(clipboard);
```

SetClipboard Parameter - clipboard

clipboard (ODClipboard *) - input
A reference to the new clipboard object for this session.

SetClipboard - Return Value

None.

SetClipboard - Parameters

clipboard (ODClipboard *) - input

A reference to the new clipboard object for this session.

None.

SetClipboard - Remarks

This method may be called only by a shell plug-in's installation function.

SetClipboard - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetDispatcher

SetDispatcher - Syntax

This method replaces the dispatcher object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODDispatcher      *dispatcher;
```

```
SetDispatcher(dispatcher);
```

SetDispatcher Parameter - dispatcher

dispatcher (ODDispatcher *) - input

A reference to the new dispatcher for this session.

SetDispatcher - Return Value

None.

SetDispatcher - Parameters

dispatcher (ODDispatcher *) - input
A reference to the new dispatcher for this session.

None.

SetDispatcher - Remarks

This method may be called only by a shell plug-in's installation function.

SetDispatcher - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetDragAndDrop

SetDragAndDrop - Syntax

This method replaces the drag-and-drop object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```



```
ODDragAndDrop      *dragAndDrop;  
SetDragAndDrop(dragAndDrop);
```

SetDragAndDrop Parameter - dragAndDrop

dragAndDrop (ODDragAndDrop *) - input
A reference to the new drag-and-drop object for this session.

SetDragAndDrop - Return Value

None.

SetDragAndDrop - Parameters

dragAndDrop (ODDragAndDrop *) - input
A reference to the new drag-and-drop object for this session.

None.

SetDragAndDrop - Remarks

This method may be called only by a shell plug-in's installation function.

SetDragAndDrop - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetInfo

SetInfo - Syntax

This method replaces the information object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>

ODInfo      *info;

SetInfo(info);
```

SetInfo Parameter - info

info (ODInfo *) - input
A reference to the new info object for this session.

SetInfo - Return Value

None.

SetInfo - Parameters

info (ODInfo *) - input
A reference to the new info object for this session.

None.

SetInfo - Remarks

This method can be called only by a shell plug-in's installation function.

SetInfo - Topics

Class:

ODSession

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)

SetLinkManager

SetLinkManager - Syntax

This method replaces the link manager object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODLinkManager      *linkManager;

SetLinkManager(linkManager);
```

SetLinkManager Parameter - linkManager

linkManager (ODLinkManager *) - input

A reference to the new link manager for this session.

SetLinkManager - Return Value

None.

SetLinkManager - Parameters

linkManager (ODLinkManager *) - input

A reference to the new link manager for this session.

None.

SetLinkManager - Remarks

This method can be called only by a shell plug-in's installation function.

SetLinkManager - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetMessageInterface

SetMessageInterface - Syntax

This method replaces the message interface for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODMessageInterface      *messageInterface;
```

```
SetMessageInterface(messageInterface);
```

SetMessageInterface Parameter - messageInterface

messageInterface (ODMessageInterface *) - input

A reference to the new message interface for this session.

SetMessageInterface - Return Value

None.

SetMessageInterface - Parameters

messageInterface (ODMessageInterface *) - input
A reference to the new message interface for this session.

None.

SetMessageInterface - Remarks

This method can be called only by a shell plug-in's installation function.

SetMessageInterface - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetNameResolver

SetNameResolver - Syntax

This method replaces the name resolver object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODNameResolver      *nameResolver;

SetNameResolver(nameResolver);
```

SetNameResolver Parameter - nameResolver

nameResolver (ODNameResolver *) - input
A reference to the new name resolver for this session.

SetNameResolver - Return Value

None.

SetNameResolver - Parameters

nameResolver (ODNameResolver *) - input
A reference to the new name resolver for this session.

None.

SetNameResolver - Remarks

This method may be called only by a shell plug-in's installation function.

SetNameResolver - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetNameSpaceManager

SetNameSpaceManager - Syntax

This method replaces the name-space manager object for this session.

```
#define INCL_ODSESSSION
#define INCL_ODAPI
#include <os2.h>

ODNameSpaceManager      *nameSpaceManager;

SetNameSpaceManager (nameSpaceManager);
```

SetNameSpaceManager Parameter - nameSpaceManager

nameSpaceManager (ODNameSpaceManager *) - input
A reference to the new name-space manager for this session.

SetNameSpaceManager - Return Value

None.

SetNameSpaceManager - Parameters

nameSpaceManager (ODNameSpaceManager *) - input
A reference to the new name-space manager for this session.

None.

SetNameSpaceManager - Remarks

This method may be called only by a shell plug-in's installation function.

SetNameSpaceManager - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetShellSemtInterface

SetShellSemtInterface - Syntax

This method is called by the document shell or container part to replace the document shell's semantic interface object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODSemanticInterface      *shellSemanticInterface;

SetShellSemtInterface(shellSemanticInterface);
```

SetShellSemtInterface Parameter - shellSemanticInterface

shellSemanticInterface (ODSemanticInterface *) - input
A reference to the new document shell's semantic interface object.

SetShellSemtInterface - Return Value

None.

SetShellSemtInterface - Parameters

shellSemanticInterface (ODSemanticInterface *) - input
A reference to the new document shell's semantic interface object.

None.

SetShellSemtInterface - Remarks

The session releases the existing semantic interface object before acquiring a new one.

This method may be called only by a shell plug-in's installation function.

For more information on semantic interfaces, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

SetShellSemtInterface - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetStorageSystem

SetStorageSystem - Syntax

This method replaces the storage system object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageSystem    *storageSystem;
```

```
SetStorageSystem(storageSystem);
```

SetStorageSystem Parameter - storageSystem

storageSystem (ODStorageSystem *) - input

A reference to the new storage system object for this session.

SetStorageSystem - Return Value

None.

SetStorageSystem - Parameters

storageSystem (ODStorageSystem *) - input

A reference to the new storage system object for this session.

None.

SetStorageSystem - Remarks

This method may be called only by a shell plug-in's installation function.

SetStorageSystem - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetToolSpaceManager (OS/2)

SetToolSpaceManager (OS/2) - Syntax

This method replaces the tool-space manager for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODToolSpaceManager      *toolSpaceMgr;
```

```
SetToolSpaceManager(toolSpaceMgr);
```

SetToolSpaceManager (OS/2) Parameter - toolSpaceMgr

toolSpaceMgr (ODToolSpaceManager *) - input
A reference to the tool-space manager.

SetToolSpaceManager (OS/2) - Return Value

None.

SetToolSpaceManager (OS/2) - Parameters

toolSpaceMgr (ODToolSpaceManager *) - input
A reference to the tool-space manager.

None.

SetToolSpaceManager (OS/2) - Remarks

This method may be called only by a shell plug-in's installation method.

SetToolSpaceManager (OS/2) - Topics

Class:
ODSession

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetTranslation

SetTranslation - Syntax

This method replaces the translation object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODTranslation      *translation;

SetTranslation(translation);
```

SetTranslation Parameter - translation

translation (ODTranslation *) - input
A reference to the new translation object for this session.

SetTranslation - Return Value

None.

SetTranslation - Parameters

translation (ODTranslation *) - input
A reference to the new translation object for this session.

None.

SetTranslation - Remarks

This method may be called only by a shell plug-in's installation function.

SetTranslation - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetUndo

SetUndo - Syntax

This method replaces the undo object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODUndo      *undo;
```

```
SetUndo(undo);
```

SetUndo Parameter - undo

undo (ODUndo *) - input

A reference to the new undo object for this session.

SetUndo - Return Value

None.

SetUndo - Parameters

undo (ODUndo *) - input

A reference to the new undo object for this session.

None.

SetUndo - Remarks

This method may be called only by a shell plug-in's installation function.

SetUndo - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetWindowState

SetWindowState - Syntax

This method replaces the window-state object for this session.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindowState      *windowState;
```

```
SetWindowState(windowState);
```

SetWindowState Parameter - windowState

windowState (ODWindowState *) - input

A reference to the new window-state object for this session.

SetWindowState - Return Value

None.

SetWindowState - Parameters

windowState (ODWindowState *) - input

A reference to the new window-state object for this session.

None.

SetWindowState - Remarks

This method may be called only by a shell plug-in's installation function.

SetWindowState - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

Tokenize

Tokenize - Syntax

This method converts the specified type string to a token.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODType      type;
ODTokenType token;
```

```
token = Tokenize(type);
```

Tokenize Parameter - type

type ([ODType](#)) - input
A type string to be converted.

Tokenize Return Value - token

token ([ODTypeToken](#)) - returns
A tokenized string representing the token corresponding to the specified type.

Tokenize - Parameters

type ([ODType](#)) - input
A type string to be converted.

token ([ODTypeToken](#)) - returns
A tokenized string representing the token corresponding to the specified type.

Tokenize - Remarks

If the specified type string has already been converted to a token (by a previous call to the Tokenize method), this method returns the corresponding token from the type/token table; otherwise, it converts the type to a token and adds a new entry to the type/token table.

Only tokens with entries in the type/token table can be converted back to type strings (by calling the [GetType](#) method).

Tokenize - Exception Handling

kODErrOutOfMemory

There is not enough memory to generate a token.

Tokenize - Related Methods

Related Methods

- [ODSession::GetType](#)
- [ODSession::RemoveEntry](#)

Tokenize - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

UniqueUpdateID

UniqueUpdateID - Syntax

This method returns a new update ID that is unique to this session and unlikely to be repeated on the network.

```
#define INCL_ODSESSION
#define INCL_ODAPI
#include <os2.h>

ODUpdateID rc;

rc = UniqueUpdateID();
```

UniqueUpdateID Return Value - rc

rc ([ODUpdateID](#)) - returns
A unique update ID.

UniqueUpdateID - Parameters

rc ([ODUpdateID](#)) - returns
A unique update ID.

UniqueUpdateID - Remarks

An update ID uniquely identifies the version of the clipboard content or linked content. The update ID values have no significance other than the context of testing them for equality; they remain valid only during the current session.

For more information on linking and link update IDs, see the chapters on storage and data transfer in the *OpenDoc Programming Guide*.

UniqueUpdateID - Topics

Class:

ODSession

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

ODSettingsExtension

Class Definition File: SETTINGS.IDL

Class Hierarchy

SOMObject

ODObject

ODExtension

ODSettingsExtension

Description

An object of the ODSettingsExtension class represents a Properties notebook that a part editor can create and display.

The Properties notebook provides access only to the standard information properties that all OpenDoc parts have. To allow the user to access properties specific to your parts, you can create a settings extension object to add additional pages to the notebook.

You can subclass ODSettingsExtension to create and insert additional pages into the Properties notebook. Once you implement it, OpenDoc accesses your settings extension object by calling your part's [AcquireExtension](#) method, which returns a reference to the extension object.

You can also remove pages from the Properties notebook by subclassing ODSettingsExtension, overriding the appropriate `Insertxxx` method, and returning `kODSettingsPageRemoved`.

For more information related to extension objects, see the class description for [ODExtension](#).

Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODSettingsExtension.

somInit

This method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODSettingsExtension, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in this settings extension object. The SOM library calls this method when this settings extension object is created. You must not do anything that could cause this method to fail. This limits you to operations such as setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this settings extension object's class specific initialization method; also see the method [InitSettingsExtension](#).

somUninit

This method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODSettingsExtension, you can override this method. Your override method does not need to call its inherited method; the

inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for this settings extension object, including any storage related to additional instance variables initialized in this settings extension object. The SOM library calls this method when this settings extension object is deleted; this method must not fail.

Release

This method decrements an object's reference count by 1; it is inherited from the [ODRefCountObject](#) class.

```
void Release ();
```

If you subclass [ODSettingsExtension](#), you can override this method to release an object and reclaim valuable resources like memory. Your override method must call its inherited method at the beginning of your implementation.

The inherited Release method decrements this settings extension object's reference count by 1. The inherited method may delete this settings extension object from memory (if this object's reference count becomes 0). OpenDoc calls this method when it no longer needs a reference to this settings extension object (for example, after the user has dismissed the Properties notebook).

Purge

This method frees memory on request; it is inherited from the [ODObject](#) class.

```
ODSize Purge (in ODSIZE size);
```

If you subclass [ODObject](#), you can override this method and should do so if it creates caches and temporary buffers. If you subclass [ODSettingsExtension](#), you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method and because you will need to compute the value returned from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

Methods

The methods defined by the [ODSettingsExtension](#) class include:

- [AddNotebookPages](#)
- [IconChanged](#)
- [InitSettingsExtension](#)
- [InsertNotebookPage](#)
- [InsertPresentationPage](#)
- [QueryIcon](#)
- [QueryInfoFacet](#)
- [QueryTitle](#)
- [RemoveNotebookPages](#)
- [ShowSettings](#)
- [TitleChanged](#)

Overridden Methods

There are currently no methods overridden by the [ODSettingsExtension](#) class.

AddNotebookPages (OS/2)

AddNotebookPages (OS/2) - Syntax

This method inserts a page into the Properties notebook.

```

#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HWND      hwndNotebook;
ODULong    flPageFlag;
ODULong    ulReserved;
ODULong    rc;

rc = AddNotebookPages(hwndNotebook, flPageFlag,
    ulReserved);

```

AddNotebookPages (OS/2) Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input
 A PM window handle to the notebook control.

AddNotebookPages (OS/2) Parameter - flPageFlag

flPageFlag ([ODULong](#)) - input
 A bit flag representing the page or group of pages to be added to the Properties notebook. This parameter can be set to one of the following values:

kODAllPage	Insert all pages into the notebook.
kODDetailsViewPage	Insert the Details View page into the notebook.
kODFile1Page	Insert the File Attributes page into the notebook.
kODFile2Page	Insert the File Comments page into the notebook.
kODFolderViewPage	Insert the Folder View page into the notebook.
kODGeneralPage	Insert the General page into the notebook.
kODIconViewPage	Insert the Icon View page into the notebook.
kODLinkPage	Insert the Link page into the notebook.
kODLinkSourcePage	Insert the Link Source page into the notebook.
kODLinkTargetPage	Insert the Link Target page into the notebook.
kODStandardPage	Insert the Standard page into the notebook.
kODTreeViewPage	Insert the Tree View page into the notebook.
kODTypePage	Insert the Type page into the notebook.
kODViewPage	Insert the View page into the notebook.

AddNotebookPages (OS/2) Parameter - ulReserved

ulReserved (ODULong) - input
Reserved value, must be 0.

AddNotebookPages (OS/2) Return Value - rc

rc (ODULong) - returns
The number of pages not added to the notebook or 0 if all pages were added.

AddNotebookPages (OS/2) - Parameters

hwndNotebook (HWND) - input
A PM window handle to the notebook control.

flPageFlag (ODULong) - input
A bit flag representing the page or group of pages to be added to the Properties notebook. This parameter can be set to one of the following values:

kODAllPage	Insert all pages into the notebook.
kODDetailsViewPage	Insert the Details View page into the notebook.
kODFile1Page	Insert the File Attributes page into the notebook.
kODFile2Page	Insert the File Comments page into the notebook.
kODFolderViewPage	Insert the Folder View page into the notebook.
kODGeneralPage	Insert the General page into the notebook.
kODIconViewPage	Insert the Icon View page into the notebook.
kODLinkPage	Insert the Link page into the notebook.
kODLinkSourcePage	Insert the Link Source page into the notebook.
kODLinkTargetPage	Insert the Link Target page into the notebook.
kODStandardPage	Insert the Standard page into the notebook.
kODTreeViewPage	Insert the Tree View page into the notebook.
kODTypePage	Insert the Type page into the notebook.
kODViewPage	Insert the View page into the notebook.

ulReserved (ODULong) - input
Reserved value, must be 0.

rc (ODULong) - returns
The number of pages not added to the notebook or 0 if all pages were added.

AddNotebookPages (OS/2) - Remarks

The base class implementation of this method inserts pages for common part properties, such as part name and part type. To add pages, the

part editor should subclass this method and call the [InsertNotebookPage](#) method after calling the parent's copy of this method.

AddNotebookPages (OS/2) - Topics

Class:

ODSettingsExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

IconChanged (OS/2)

IconChanged (OS/2) - Syntax

This method notifies the part when the part's icon is changed through the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
HPOINTER    hptrNewIcon;
```

```
IconChanged(hptrNewIcon);
```

IconChanged (OS/2) Parameter - hptrNewIcon

hptrNewIcon ([HPOINTER](#)) - input

A handle to the part's icon.

IconChanged (OS/2) - Return Value

None.

IconChanged (OS/2) - Parameters

hptrNewIcon ([HPOINTER](#)) - input
A handle to the part's icon.

None.

IconChanged (OS/2) - Remarks

The storage associated with *hptrNewIcon* is owned by the notebook control and should not be freed or modified by the part.

IconChanged (OS/2) - Override Policy

If you subclass [ODSettingsExtension](#), you must override this method to receive notification when the part's icon is changed through the Properties notebook.

IconChanged (OS/2) - Related Methods

Related Methods

- [ODSettingsExtension::QueryIcon](#)
-

IconChanged (OS/2) - Topics

Class:

[ODSettingsExtension](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Override Policy](#)
[Related Methods](#)

InitSettingsExtension

InitSettingsExtension - Syntax

This method should initialize this settings extension object.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

ODPart      *owner;

InitSettingsExtension(owner);
```

InitSettingsExtension Parameter - owner

owner (ODPart *) - input
A reference to this settings extension's base object.

InitSettingsExtension - Return Value

None.

InitSettingsExtension - Parameters

owner (ODPart *) - input
A reference to this settings extension's base object.

None.

InitSettingsExtension - Remarks

This method is not called directly to initialize this settings extension object, but is called by a subclass-specific initialization method. By convention, every subclass of [ODSettingsExtension](#) should have an override method that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the inherited `InitSettingsExtension` method. The override method should call the inherited `InitSettingsExtension` method at the beginning of its implementation. The inherited `InitSettingsExtension` method in turn calls the [InitExtension](#) method associated with this settings extension's base object ([ODExtension](#)) to prepare this settings extension for use.

If you subclass [ODSettingsExtension](#), your subclass-specific initialization method, rather than its `somInit` method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your settings extensin.

InitSettingsExtension - Override Policy

If you subclass [ODSettingsExtension](#), you must override this method. Your override method must call its inherited method at the beginning of your implementation.

InitSettingsExtension - Related Methods

Related Methods

- [ODExtension::InitExtension](#)
-

InitSettingsExtension - Topics

Class:

ODSettingsExtension

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Override Policy](#)
[Related Methods](#)

InsertNotebookPage (OS/2)

InsertNotebookPage (OS/2) - Syntax

This method inserts a notebook page into the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HWND      hwndNotebook;
PPAGEINFO pPageInfo;
ODULong    PageID;

PageID = InsertNotebookPage(hwndNotebook,
                             pPageInfo);
```

InsertNotebookPage (OS/2) Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

InsertNotebookPage (OS/2) Parameter - pPageInfo

pPageInfo ([PPAGEINFO](#)) - input
A pointer to the notebook page information.

InsertNotebookPage (OS/2) Return Value - PageID

PageID ([ODULong](#)) - returns
The page ID of the inserted page.

InsertNotebookPage (OS/2) - Parameters

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

pPageInfo ([PPAGEINFO](#)) - input
A pointer to the notebook page information.

PageID ([ODULong](#)) - returns
The page ID of the inserted page.

InsertNotebookPage (OS/2) - Remarks

This method can be called only during the processing of the [AddNotebookPages](#) method.

InsertNotebookPage (OS/2) - Exception Handling

kODErrOutOfMemory

There is not enough memory to create the page.

kODErrPMEError

A PM API returned an error indication. The WinGetLastError function can be used to obtain the error code.

InsertNotebookPage (OS/2) - Topics

Class:
ODSettingsExtension

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

InsertPresentationPage (OS/2)

InsertPresentationPage (OS/2) - Syntax

This method inserts the *Presentation* page into the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HWND          hwndNotebook;
ODTypeList    *presentationList;
ODULong       PageId;

PageId = InsertPresentationPage(hwndNotebook,
                                presentationList);
```

InsertPresentationPage (OS/2) Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

InsertPresentationPage (OS/2) Parameter - presentationList

presentationList (ODTypeList *) - input
A reference to a type-list object.

InsertPresentationPage (OS/2) Return Value - PageId

PageId ([ODULong](#)) - returns
The ID of the inserted page. A return value of 0 indicates that an error occurred.

InsertPresentationPage (OS/2) - Parameters

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

presentationList (ODTypeList *) - input
A reference to a type-list object.

PageId ([ODULong](#)) - returns
The ID of the inserted page. A return value of 0 indicates that an error occurred.

InsertPresentationPage (OS/2) - Remarks

The *Presentation* page allows the user to select the part kind and preferred part editor for the part.

This method must be called only from within an override of the [AddNotebookPages](#) method.

InsertPresentationPage (OS/2) - Exception Handling

kODErrorOutOfMemory

There is not enough memory to create the page.

kODErrPMEError

A PM API returned an error indication. The WinGetLastError function can be called to obtain the error code.

InsertPresentationPage (OS/2) - Topics

Class:
ODSettingsExtension

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

QueryIcon (OS/2)

QueryIcon (OS/2) - Syntax

This method returns the icon of the part to be displayed on the *General* page of the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HPOINTER icon;

icon = QueryIcon();
```

QueryIcon (OS/2) Return Value - icon

icon ([HPOINTER](#)) - returns
A pointer to the current icon.

QueryIcon (OS/2) - Parameters

icon ([HPOINTER](#)) - returns
A pointer to the current icon.

QueryIcon (OS/2) - Remarks

The new icon is either the icon which was last set by the user or the icon set by a call to [IconChanged](#), depending on which event occurred last. If neither event has occurred, the default icon is displayed on the *General* page.

QueryIcon (OS/2) - Related Methods

Related Methods

- [ODSettingsExtension::IconChanged](#)

QueryIcon (OS/2) - Topics

Class:
[ODSettingsExtension](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

QueryInfoFacet (OS/2)

QueryInfoFacet (OS/2) - Syntax

This method returns the facet information of the specified notebook control.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HWND      hwndNotebook;
ODFacet    *facet;

facet = QueryInfoFacet (hwndNotebook);
```

QueryInfoFacet (OS/2) Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

QueryInfoFacet (OS/2) Return Value - facet

facet (ODFacet *) - returns
The facet on which this notebook was opened.

QueryInfoFacet (OS/2) - Parameters

hwndNotebook ([HWND](#)) - input
A PM window handle to the notebook control.

facet (ODFacet *) - returns
The facet on which this notebook was opened.

QueryInfoFacet (OS/2) - Topics

Class:

ODSettingsExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

QueryTitle (OS/2)

QueryTitle (OS/2) - Syntax

This method returns the title of the part to be displayed on the *General* page of the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

char      *newTitle;

newTitle = QueryTitle();
```

QueryTitle (OS/2) Return Value - newTitle

newTitle (char *) - returns
A pointer to the current title.

QueryTitle (OS/2) - Parameters

newTitle (char *) - returns
A pointer to the current title.

QueryTitle (OS/2) - Remarks

This title is either the last title set by the user or the title set by a call to the SetTitle method. If neither event has occurred, no title is displayed on the *General* page.

QueryTitle (OS/2) - Related Methods

Related Methods

- [ODSettingsExtension::TitleChanged](#)
-

QueryTitle (OS/2) - Topics

Class:

[ODSettingsExtension](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

RemoveNotebookPages (OS/2)

RemoveNotebookPages (OS/2) - Syntax

This method removes a page from the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

HWND      hwndNotebook;
ODULong    flPageFlag;
ODULong     rc;

rc = RemoveNotebookPages(hwndNotebook, flPageFlag);
```

RemoveNotebookPages (OS/2) Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input

A PM window handle to the notebook control.

RemoveNotebookPages (OS/2) Parameter - flPageFlag

flPageFlag (ODULong) - input

A bit flag representing the page or group of pages to be removed from the Properties notebook. This parameter can be set to one of the following values:

kODAllPage	Insert all pages into the notebook.
kODDetailsViewPage	Insert the Details View page into the notebook.
kODFile1Page	Insert the File Attributes page into the notebook.
kODFile2Page	Insert the File Comments page into the notebook.
kODFolderViewPage	Insert the Folder View page into the notebook.
kODGeneralPage	Insert the General page into the notebook.
kODIconViewPage	Insert the Icon View page into the notebook.
kODLinkPage	Insert the Link page into the notebook.
kODLinkSourcePage	Insert the Link Source page into the notebook.
kODLinkTargetPage	Insert the Link Target page into the notebook.
kODStandardPage	Insert the Standard page into the notebook.
kODTreeViewPage	Insert the Tree View page into the notebook.
kODTypePage	Insert the Type page into the notebook.
kODViewPage	Insert the View page into the notebook.

RemoveNotebookPages (OS/2) Return Value - rc

rc (ODULong) - returns

The number of pages not added to the notebook or 0 if all pages were added.

RemoveNotebookPages (OS/2) - Parameters

hwndNotebook (HWND) - input

A PM window handle to the notebook control.

flPageFlag (ODULong) - input

A bit flag representing the page or group of pages to be removed from the Properties notebook. This parameter can be set to one of the following values:

kODAllPage	Insert all pages into the notebook.
kODDetailsViewPage	Insert the Details View page into the notebook.

kODFile1Page	Insert the File Attributes page into the notebook.
kODFile2Page	Insert the File Comments page into the notebook.
kODFolderViewPage	Insert the Folder View page into the notebook.
kODGeneralPage	Insert the General page into the notebook.
kODIconViewPage	Insert the Icon View page into the notebook.
kODLinkPage	Insert the Link page into the notebook.
kODLinkSourcePage	Insert the Link Source page into the notebook.
kODLinkTargetPage	Insert the Link Target page into the notebook.
kODStandardPage	Insert the Standard page into the notebook.
kODTreeViewPage	Insert the Tree View page into the notebook.
kODTypePage	Insert the Type page into the notebook.
kODViewPage	Insert the View page into the notebook.

rc ([ODULong](#)) - returns
The number of pages not added to the notebook or 0 if all pages were added.

RemoveNotebookPages (OS/2) - Topics

Class:
ODSettingsExtension

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ShowSettings

ShowSettings - Syntax

This method displays the Properties notebook so the user can edit part-specific properties.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;

ShowSettings(facet);
```

ShowSettings Parameter - facet

facet (ODFacet *) - input

A reference to a facet that indicates the window (and indirectly the monitor) in which to display the Properties notebook. The facet's frame indicates which frame of your part's properties should be displayed and edited if your part has part-specific properties that are then also frame specific.

ShowSettings - Return Value

None.

ShowSettings - Parameters

facet (ODFacet *) - input

A reference to a facet that indicates the window (and indirectly the monitor) in which to display the Properties notebook. The facet's frame indicates which frame of your part's properties should be displayed and edited if your part has part-specific properties that are then also frame specific.

None.

ShowSettings - Override Policy

If you subclass [ODSettingsExtension](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

ShowSettings - Topics

Class:

ODSettingsExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Override Policy](#)

TitleChanged (OS/2)

TitleChanged (OS/2) - Syntax

This method notifies the part when the part's title is changed through the Properties notebook.

```
#define INCL_ODSETTINGSEXTENSION
#define INCL_ODAPI
#include <os2.h>

char      *newTitle;

TitleChanged(newTitle);
```

TitleChanged (OS/2) Parameter - newTitle

newTitle (char *) - input
A pointer to the new title.

TitleChanged (OS/2) - Return Value

None.

TitleChanged (OS/2) - Parameters

newTitle (char *) - input
A pointer to the new title.

None.

TitleChanged (OS/2) - Remarks

The storage pointed to by the *newTitle* parameter is owned by the notebook control and should not be freed modified by the part.

TitleChanged (OS/2) - Related Methods

Related Methods

- [ODSettingsExtension::QueryTitle](#)

TitleChanged (OS/2) - Topics

Class:

ODSettingsExtension

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

ODShape

Class Definition File: SHAPE.IDL

Class Hierarchy

SOMObject
 ODObject
 ODRefCntObject
 ODBaseShape
 ODShape

Description

An object of the ODShape class represents a geometric shape that OpenDoc uses to manage the display of parts.

Shape objects encapsulate geometric shapes. The simplest shape is an *empty* shape, which occupies no area.

Your part creates an empty shape by calling the [CreateShape](#) method of a frame, the [CreateShape](#) method of a facet, or the [NewShape](#) method of an existing shape. Your part can create a copy of an existing shape by calling that shape's [Copy](#) method.

Frames and facets use shape objects for frame negotiation, clipping, and hit-testing. Shape objects represent four kinds of shapes used for those purposes:

Active shape	Defines the area of a facet within which the embedded part responds to mouse events.
Clip shape	Defines the area of a facet in which drawing can occur; it is the area not obscured by overlapping content of the containing part.
Frame shape	Defines the area that the containing part delegates to an embedded part's display frame.
Used shape	Defines the area of an embedded part's frame that has actual content to display; the containing part is free to display its own content within the embedded frame, but it must remain outside the used shape area.

Typically, your part creates a shape object by calling the shape object's [Copy](#) method; however, your part can also create a shape by calling the [NewShape](#) and [CreateShape](#) methods. Both of these methods return a shape object.

Geometric Representation

The geometric representation of a shape is a description of the area it encompasses in its coordinate space. Shape objects can maintain their geometric representation as a polygon or in a platform specific manner (such as an OS/2 GPI region). A polygon representation of a shape can always be converted to a platform specific representation. If a shape does not have a polygonal representation, one can always be generated from the platform specific representation; however, the resulting polygon will only be an approximation of the shape. For example, on OS/2, if a polygon is generated from a region that represents a triangular shape, the resulting polygon will have many more than 3 vertices. That is because the polygon is generated by tracing the exposed edges of the region rectangles, with one vertex for each corner of the region rectangle.

Geometry Mode

The *geometry mode* of a shape object which specifies whether the shape is required to maintain its geometric (polygonal) representation. The geometry mode has three possible values:

Preserve geometry	Indicates that the shape must maintain its polygonal representation for as long as possible. A shape's polygonal representation is lost if the shape is combined with (that is, unioned with, subtracted from, or intersected with) a shape that does not have polygonal representation. The shape's polygonal representation is replaced by the area that results from combining the two shapes. This is the default.
Lose geometry	Indicates that the shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, but at the expense of accuracy and persistent storage capability. For example, on OS/2, the shape's geometric representation can be represented by a GPI region rather than a polygon.
Needs geometry	Indicates that the shape must always maintain its polygonal representation. The shape cannot be unioned with, subtracted from, or intersected with a shape that does not have a polygonal representation. If you attempt to do so, the shape raises an exception and does not notify its geometric representation.

Geometric Operations

Shape objects support geometric operations such as union, intersection, and difference. A platform-specific shape-manipulation engine may be used to perform these operations. As a consequence, the results of certain operations, such as test for equality, difference, union, and intersection, may vary slightly from platform to platform; these cross-platform differences should be apparent only at the pixel level.

For more information about frame shapes and used shapes, see the description of the [ODFrame](#) class. For information about clip shapes and active shapes, see the description of the [ODFacet](#) class. For more information on graphics systems available on OS/2, see the *Graphics Programming Interface Programming Guide*.

Methods

The methods defined by the ODShape class include:

Creating Shapes [Copy NewShape](#)

Manipulating Shape Geometry

- [CopyFrom](#)
- [CopyPolygon](#)
- [CopyRegion](#)
- [GetBoundingBox](#)
- [GetPlatformShape](#)
- [GetRegion](#)
- [ReadShape](#)
- [Reset](#)
- [SetPlatformShape](#)
- [SetPolygon](#)
- [SetRectangle](#)
- [SetRegion](#)
- [WriteShape](#)

Testing Shapes

- [ContainsPoint](#)
- [HasGeometry](#)
- [IsEmpty](#)
- [IsRectangular](#)
- [IsSameAs](#)

Manipulating Geometry Mode

- [GetGeometryMode](#)
- [SetGeometryMode](#)

Performing Geometric Operations

- [Intersect](#)
- [InverseTransform](#)
- [Outset](#)
- [Subtract](#)
- [Transform](#)
- [Union](#)

Overridden Methods

There are currently no methods overridden by the ODShape class.

ContainsPoint

ContainsPoint - Syntax

This method indicates whether the specified point is within the area of this shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;
ODBoolean     rv;

rv = ContainsPoint(point);
```

ContainsPoint Parameter - point

point (ODPoint *) - input
The point to test, expressed in this shape's coordinate space.

ContainsPoint Return Value - rv

rv (ODBoolean) - returns
A flag indicating whether the point is within this shape's area.

kODTrue	The point is within this shape's area.
kODFalse	The point is outside of this shape's area.

ContainsPoint - Parameters

point (ODPoint *) - input
The point to test, expressed in this shape's coordinate space.

rv (ODBoolean) - returns
A flag indicating whether the point is within this shape's area.

kODTrue

kODFalse

The point is within this shape's area.

The point is outside of this shape's area.

ContainsPoint - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

Copy

Copy - Syntax

This method creates a new shape object that is a copy of this shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *rv;
```

```
rv = Copy();
```

Copy Return Value - rv

rv (ODShape *) - returns

A reference to the newly created shape.

Copy - Parameters

rv (ODShape *) - returns

A reference to the newly created shape.

Copy - Remarks

This method creates a shape that the copy is going to be placed into.

The new shape does not share any data with this shape; therefore, you can modify each of the shapes independently. This method automatically deletes the new shape if the copy operation fails.

This method initializes the reference count of the returned shape. When you have finished using that shape, you should call its [Release](#) method.

Copy - Exception Handling

kODErrOutOfMemory

There is not enough memory to copy this shape.

Copy - Related Methods

Related Methods

- [ODShape::CopyFrom](#)
-

Copy - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CopyFrom

CopyFrom - Syntax

This method modifies this shape to make it equivalent to the specified source shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *sourceShape;

CopyFrom(sourceShape);
```

CopyFrom Parameter - sourceShape

sourceShape (ODShape *) - input
A reference to the source shape.

CopyFrom - Return Value

None.

CopyFrom - Parameters

sourceShape (ODShape *) - input
A reference to the source shape.

None.

CopyFrom - Remarks

After this method executes successfully, this shape and the source shape do not share any data; therefore, you can modify each of them independently.

CopyFrom - Exception Handling

kODErrOutOfMemory

There is not enough memory to update this shape.

CopyFrom - Related Methods

Related Methods

- [ODShape::Copy](#)

CopyFrom - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

CopyPolygon (OS/2)

CopyPolygon (OS/2) - Syntax

This method returns a copy of this shape's geometric representation, expressed as a polygon.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
ODPolygon      *copy;
```

```
CopyPolygon (copy);
```

CopyPolygon (OS/2) Parameter - copy

copy ([ODPolygon](#) *) - output

A structure whose fields are set to the represent a polygon that describes this shape's geometric representation, or an empty polygon if this shape's geometric representation cannot be represented by a polygon.

CopyPolygon (OS/2) - Return Value

None.

CopyPolygon (OS/2) - Parameters

copy ([ODPolygon *](#)) - output

A structure whose fields are set to the represent a polygon that describes this shape's geometric representation, or an empty polygon if this shape's geometric representation cannot be represented by a polygon.

None.

CopyPolygon (OS/2) - Remarks

To check whether the shape's geometric representation can be described by a polygon, call the [HasGeometry](#) method. Note that some geometric representations, such as curves, can only be approximated by a polygon.

The polygon returned in the *copy* output parameter is not owned by this shape; you are allowed to [modify it](#). When you no longer need the polygon, you should deallocate its storage.

CopyPolygon (OS/2) - Exception Handling

kODErrNoShapeGeometry

This shape has no geometric representation, so it cannot be described as a polygon.

kODErrOutOfMemory

There is not enough memory to copy this shape's geometric representation.

CopyPolygon (OS/2) - Related Methods

Related Methods

- [ODShape::HasGeometry](#)
-

CopyPolygon (OS/2) - Topics

Class:

ODShape

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

CopyRegion (OS/2)

CopyRegion (OS/2) - Syntax

This method returns a GPI region equivalent to the shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODRgnHandle    rv;

rv = CopyRegion();
```

CopyRegion (OS/2) Return Value - rv

rv (ODRgnHandle) - returns
A handle to the GPI region.

CopyRegion (OS/2) - Parameters

rv (ODRgnHandle) - returns
A handle to the GPI region.

CopyRegion (OS/2) - Remarks

Prior to calling this method, you must transform the shape to the coordinate space of the device that you intend to use the region for.

This method creates a new region and copies the shapes into it. The region is owned by the caller, and the caller is responsible for destroying the region when it is done using it.

CopyRegion (OS/2) - Topics

Class:
ODShape

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetBoundingBox

GetBoundingBox - Syntax

This method returns, in the specified structure, the smallest rectangle that surrounds this shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODRect      *bounds;

GetBoundingBox(bounds);
```

GetBoundingBox Parameter - bounds

bounds ([ODRect](#) *) - output
A rectangle describing this shape's bounding box.

GetBoundingBox - Return Value

None.

GetBoundingBox - Parameters

bounds ([ODRect](#) *) - output
A rectangle describing this shape's bounding box.

None.

GetBoundingBox - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetGeometryMode

GetGeometryMode - Syntax

This method returns the current geometry mode of this shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODGeometryMode    rv;

rv = GetGeometryMode();
```

GetGeometryMode Return Value - rv

rv ([ODGeometryMode](#)) - returns

The geometry mode for this shape. This parameter can return one of the following values:

kODLoseGeometry

The shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, at the expense of accuracy and persistent storage capability. A facet's clip shape generally has this mode.

kODNeedsGeometry

The shape must maintain its polygonal presentation. A facet's frame shape and used shape has this mode because they are stored persistently in polygonal form.

kODPreserveGeometry

The shape must maintain its polygonal representation for as long as possible. This is the default value.

GetGeometryMode - Parameters

rv ([ODGeometryMode](#)) - returns

The geometry mode for this shape. This parameter can return one of the following values:

kODLoseGeometry

The shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, at the expense of accuracy and persistent storage capability. A facet's clip shape generally has this mode.

kODNeedsGeometry

The shape must maintain its polygonal presentation. A facet's frame shape and used shape has this mode because they are stored persistently in polygonal form.

kODPreserveGeometry

The shape must maintain its polygonal representation for as long as possible. This is the default value.

GetGeometryMode - Remarks

The geometry mode of a shape object specifies whether the shape is required to maintain its geometric (polygonal) representation.

GetGeometryMode - Related Methods

Related Methods

- [ODShape::SetGeometryMode](#)

GetGeometryMode - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

GetPlatformShape

GetPlatformShape - Syntax

This method returns a graphics-system-specific data structure representing this shape.


```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    graphicsSystem;
ODPlatformShape     rv;

rv = GetPlatformShape(graphicsSystem);
```

GetPlatformShape Parameter - graphicsSystem

graphicsSystem ([ODGraphicsSystem](#)) - input

The graphics system ID identifying the graphics system you want to use for this shape. Valid graphics systems are platform-dependent. For OS/2, this value must be set to kODGPI.

GetPlatformShape Return Value - rv

rv ([ODPlatformShape](#)) - returns

The requested graphics-system-specific shape. Before using the returned value, you must must cast it to a valid graphics system type. On OS/2, this method returns a GPI region handle.

GetPlatformShape - Parameters

graphicsSystem ([ODGraphicsSystem](#)) - input

The graphics system ID identifying the graphics system you want to use for this shape. Valid graphics systems are platform-dependent. For OS/2, this value must be set to kODGPI.

rv ([ODPlatformShape](#)) - returns

The requested graphics-system-specific shape. Before using the returned value, you must must cast it to a valid graphics system type. On OS/2, this method returns a GPI region handle.

GetPlatformShape - Remarks

The returned region handle is owned by the shape object. You must not modify or destroy the region.

GetPlatformShape - Exception Handling

kODErrInvalidGraphicsSystem

This implementation of OpenDoc does not support the specified graphics system, or that graphics system is not installed or available.

GetPlatformShape - Related Methods

Related Methods

- [ODShape::SetPlatformShape](#)
-

GetPlatformShape - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetRegion (OS/2)

GetRegion (OS/2) - Syntax

This method returns a GPI region equivalent to the shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODRgnHandle    rv;

rv = GetRegion();
```

GetRegion (OS/2) Return Value - rv

rv (ODRgnHandle) - returns
A handle to the GPI region.

GetRegion (OS/2) - Parameters

rv (ODRgnHandle) - returns
A handle to the GPI region.

GetRegion (OS/2) - Remarks

The returned region handle is owned by the shape object. You must not modify or destroy the region.

GetRegion (OS/2) - Topics

Class:
ODShape

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

HasGeometry

HasGeometry - Syntax

This method indicates whether the geometric representation of this shape object can be described by a polygon.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = HasGeometry();
```

HasGeometry Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this shape's geometric representation can be described by a polygon.

On OS/2, this method always returns `KODTrue` because a polygonal representation can always be generated from a nonpolygonal representation.

HasGeometry - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this shape's geometric representation can be described by a polygon.

On OS/2, this method always returns `KODTrue` because a polygonal representation can always be generated from a nonpolygonal representation.

HasGeometry - Remarks

Your part can use this method, before calling the [CopyPolygon](#) method, to verify that a shape's geometric representation can be described by a polygon.

HasGeometry - Related Methods

Related Methods

- [ODShape::CopyPolygon](#)
- [ODShape::SetGeometryMode](#)

HasGeometry - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

Intersect

Intersect - Syntax

This method modifies this shape by intersecting it with the specified shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *sectShape;
ODShape      *rv;

rv = Intersect(sectShape);
```

Intersect Parameter - sectShape

sectShape (ODShape *) - input
A reference to the shape to be intersected with this shape.

Intersect Return Value - rv

rv (ODShape *) - returns
A reference to this shape after the intersection operation.

Intersect - Parameters

sectShape (ODShape *) - input
A reference to the shape to be intersected with this shape.

rv (ODShape *) - returns
A reference to this shape after the intersection operation.

Intersect - Remarks

After this method executes successfully, this shape is equivalent to the intersected area.

Intersect - Exception Handling

kODErrNoShapeGeometry

The geometry mode of this shape is kODNeedsGeometry, but the other shape has no geometric representation.

kODErrOutOfMemory

There is not enough memory to intersect the shapes.

Intersect - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

InverseTransform

InverseTransform - Syntax

This method modifies this shape by applying the inverse of the specified transform to it.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODTransform      *ttransform;
ODShape          *rv;

rv = InverseTransform(transform);
```

InverseTransform Parameter - transform

transform (ODTransform *) - input

A reference to the transform whose inverse is to be applied to this shape.

InverseTransform Return Value - rv

rv (ODShape *) - returns

A reference to this shape after it has been transformed.

InverseTransform - Parameters

transform (ODTransform *) - input
A reference to the transform whose inverse is to be applied to this shape.

rv (ODShape *) - returns
A reference to this shape after it has been transformed.

InverseTransform - Remarks

This method is the inverse operation of the [Transform](#) method.

InverseTransform - Exception Handling

kODErrNoShapeGeometry	This shape does not have enough geometric information to be transformed with the inverse of the specified transform.
kODErrTransformErr	The tranform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

InverseTransform - Related Methods

Related Methods

- [ODShape::Transform](#)
- [ODTransform::Invert](#)

InverseTransform - Topics

Class:
ODShape

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

IsEmpty

IsEmpty - Syntax

This method indicates whether this shape is empty (occupies no area).

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsEmpty();
```

IsEmpty Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the shape is empty.

kODTrue	The shape is empty.
kODFalse	The shape is not empty.

IsEmpty - Parameters

rv ([ODBoolean](#)) - returns
A flag indicating whether the shape is empty.

kODTrue	The shape is empty.
kODFalse	The shape is not empty.

IsEmpty - Remarks

A shape that occupies no area is known as an empty shape. An empty is typically described by the empty rectangle (0,0,0,0). Alternatively, it can be described by a polygon with 0 contours.

IsEmpty - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

IsRectangular

IsRectangular - Syntax

This method indicates whether this shape can be described by a rectangle.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsRectangular();
```

IsRectangular Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the this shape is rectangular.

kODTrue

The shape is rectangular.

kODFalse

The shape is not rectangular.

IsRectangular - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether the this shape is rectangular.

kODTrue

The shape is rectangular.

kODFalse

The shape is not rectangular.

IsRectangular - Remarks

Empty shapes are considered to be rectangular; they can be described by the empty rectangle (0,0,0,0).

IsRectangular - Related Methods

Related Methods

- [ODShape::GetBoundingBox](#)
 - [ODShape::SetRegion](#)
-

IsRectangular - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

IsSameAs

IsSameAs - Syntax

This method indicates whether this shape is identical to the specified shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *compareShape;
ODBoolean     rv;

rv = IsSameAs (compareShape);
```

IsSameAs Parameter - compareShape

compareShape (ODShape *) - input

A reference to the shape to be compared with this shape.

IsSameAs Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the shapes are equivalent.

kODTrue

The shapes are equivalent.

kODFalse

The shapes are not equivalent.

IsSameAs - Parameters

compareShape (ODShape *) - input

A reference to the shape to be compared with this shape.

rv ([ODBoolean](#)) - returns

A flag indicating whether the shapes are equivalent.

kODTrue

The shapes are equivalent.

kODFalse

The shapes are not equivalent.

IsSameAs - Remarks

If both shapes have polygonal representations, they are equivalent if they consist of the same contours. The order of the contours does not matter. Each of the corresponding contours in the two polygons must have their vertices listed in the same direction (clockwise or counterclockwise).

If this shape is described only by a platform shape (such as a region), the platform graphics system is used to determine equality; typically, the *compareShape* parameter must be a platform shape of the same type.

IsSameAs - Exception Handling

kODErrOutOfMemory

There is not enough memory to compare the two shapes.

IsSameAs - Topics

Class:

ODShape

Select an item:

NewShape

NewShape - Syntax

This method creates a new empty shape object.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *rv;

rv = NewShape();
```

NewShape Return Value - rv

rv (ODShape *) - returns
A reference to the newly created shape or KODNULL if an error occurred.

NewShape - Parameters

rv (ODShape *) - returns
A reference to the newly created shape or KODNULL if an error occurred.

NewShape - Remarks

This method initializes the reference count of the returned shape. When you have finished using that shape, you should call its [Release](#) method.

NewShape - Exception Handling

kODErrOutOfMemory

There is not enough memory to create a new shape.

NewShape - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

Outset

Outset - Syntax

This method modifies this shape by moving its boundary outwards-away from its interior-by the specified distance.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODCoordinate distance;
ODShape      *rv;

rv = Outset(distance);
```

Outset Parameter - distance

distance ([ODCoordinate](#)) - input

The distance (expressed in the shape's coordinate system) to move the shape's outline.

Outset Return Value - rv

rv (ODShape *) - returns

A reference to this shape after the outset operation.

Outset - Parameters

distance ([ODCoordinate](#)) - input

The distance (expressed in the shape's coordinate system) to move the shape's outline.

rv (ODShape *) - returns

A reference to this shape after the outset operation.

Outset - Remarks

This method is typically used to create a border around a shape. To do this, copy the original shape, outset the copy, then subtract the original shape from the copy.

To inset a shape (move the boundary inwards), the call this method with a negative distance.

Outset - Exception Handling

kODErrOutOfMemory

There is not enough memory to complete the operation. The shape will be unmodified.

Outset - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

ReadShape

ReadShape - Syntax

This method reads shape data from the specified storage unit into this shape.

```

#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;
ODShape            *rv;

rv = ReadShape(storageUnit);

```

ReadShape Parameter - storageUnit

storageUnit (ODStorageUnit *) - input
A reference to the storage unit from which data is to be read.

ReadShape Return Value - rv

rv (ODShape *) - returns
A reference to this shape.

ReadShape - Parameters

storageUnit (ODStorageUnit *) - input
A reference to the storage unit from which data is to be read.

rv (ODShape *) - returns
A reference to this shape.

ReadShape - Remarks

Before calling this method, you must focus the storage unit to the property that contains the shape data.

If the focused property contains a value of type kODPolygon, the method reads the shape data from that value. Otherwise, if the focused property contains a value of a platform-specific shape value type, this method reads the shape data from that value.

ReadShape - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or a value.

ReadShape - Related Methods

Related Methods

- [ODShape::WriteShape](#)

ReadShape - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Reset

Reset - Syntax

This method replaces this shape's geometric representation with an empty area.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
Reset ();
```

Reset - Return Value

None.

Reset - Parameters

None.

Reset - Remarks

Except for shapes created by the [Copy](#) method, newly created shapes start out as empty shapes; this method changes a shape back to the initial state.

The effect of this method is the same as replacing this shape with an empty rectangle by using the [SetRectangle](#) method with the empty rectangle (0,0,0,0); however, the Reset method is slightly more efficient.

Reset - Related Methods

Related Methods

- [ODShape::SetRectangle](#)
-

Reset - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

SetGeometryMode

SetGeometryMode - Syntax

This method sets the geometry mode of this shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>
```

```
ODGeometryMode    mode;
```

SetGeometryMode (mode) ;

SetGeometryMode Parameter - mode

mode ([ODGeometryMode](#)) - input

The geometry mode for this shape.

kODLoseGeometry

The shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, at the expense of accuracy and persistent storage capability. A facet's clip shape generally has this mode.

kODNeedsGeometry

The shape must maintain its polygonal presentation. A facet's frame shape and used shape has this mode because they are stored persistently in polygonal form.

kODPreserveGeometry

The shape must maintain its polygonal representation for as long as possible. This is the default value.

SetGeometryMode - Return Value

None.

SetGeometryMode - Parameters

mode ([ODGeometryMode](#)) - input

The geometry mode for this shape.

kODLoseGeometry

The shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, at the expense of accuracy and persistent storage capability. A facet's clip shape generally has this mode.

kODNeedsGeometry

The shape must maintain its polygonal presentation. A facet's frame shape and used shape has this mode because they are stored persistently in polygonal form.

kODPreserveGeometry

The shape must maintain its polygonal representation for as long as possible. This is the default value.

None.

SetGeometryMode - Remarks

The geometry mode of a shape object specifies whether the shape is required to maintain its geometric (polygonal) representation.

SetGeometryMode - Exception Handling

kODErrNoShapeGeometry

The specified geometry mode is kODNeedsGeometry, but the shape has no geometry.

SetGeometryMode - Related Methods

Related Methods

- [ODShape::GetGeometryMode](#)
-

SetGeometryMode - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetPlatformShape

SetPlatformShape - Syntax

This method modifies this shape to make it equivalent to the specified graphics-system-specific shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    graphicsSystem;
ODPlatformShape     platformShape;

SetPlatformShape(graphicsSystem, platformShape);
```

SetPlatformShape Parameter - graphicsSystem

graphicsSystem ([ODGraphicsSystem](#)) - input

A graphics system for which you are specifying a shape. Valid graphics systems are platform-dependent. For OS/2, this parameter must be set to kODGPI.

SetPlatformShape Parameter - platformShape

platformShape ([ODPlatformShape](#)) - input

The valid graphics-system-specific shape. Valid values are graphics-system-dependent. On OS/2, this is a region handle ([HRGN](#))

SetPlatformShape - Return Value

None.

SetPlatformShape - Parameters

graphicsSystem ([ODGraphicsSystem](#)) - input

A graphics system for which you are specifying a shape. Valid graphics systems are platform-dependent. For OS/2, this parameter must be set to kODGPI.

platformShape ([ODPlatformShape](#)) - input

The valid graphics-system-specific shape. Valid values are graphics-system-dependent. On OS/2, this is a region handle ([HRGN](#))

None.

SetPlatformShape - Remarks

After this method executes successfully, the region is owned by the shape object. The caller must not modify or destroy the region.

SetPlatformShape - Exception Handling

kODErrInvalidGraphicsSystem

The implementation of OpenDoc does not support the specified graphics system, or that graphics system is not installed or available.

SetPlatformShape - Related Methods

Related Methods

- [ODShape::GetPlatformShape](#)
-

SetPlatformShape - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

SetPolygon

SetPolygon - Syntax

This method modifies this shape to make it equivalent to the specified polygon.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODPolygon      *polygon;
ODShape        *rv;

rv = SetPolygon(polygon);
```

SetPolygon Parameter - polygon

polygon ([ODPolygon *](#)) - input
A valid polygon.

SetPolygon Return Value - rv

rv (ODShape *) - returns

A reference to this shape after its has been changed to be equivalent to the specified polygon.

SetPolygon - Parameters

polygon ([ODPolygon *](#)) - input

A valid polygon.

rv (ODShape *) - returns

A reference to this shape after its has been changed to be equivalent to the specified polygon.

SetPolygon - Remarks

After this method executes successfully, you are still responsible for deleting the original polygon; this shape does not use or modify it.

If the specified polygon is self-intersecting, certain methods, such as [Intersect](#), [Outset](#), [Subtract](#), and [Union](#) simplify the polygon by removing the areas of self intersection.

SetPolygon - Related Methods

Related Methods

- [ODShape::CopyPolygon](#)
-

SetPolygon - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

SetRectangle

SetRectangle - Syntax

This method modifies this shape to make it equivalent to the specified rectangle.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODRect      *rect;
ODShape      *rv;

rv = SetRectangle(rect);
```

SetRectangle Parameter - rect

rect ([ODRect](#) *) - input
A valid rectangle.

SetRectangle Return Value - rv

rv ([ODShape](#) *) - returns
A reference to this shape after its has beenn changed to be equivalent to the specified rectangle.

SetRectangle - Parameters

rect ([ODRect](#) *) - input
A valid rectangle.

rv ([ODShape](#) *) - returns
A reference to this shape after its has beenn changed to be equivalent to the specified rectangle.

SetRectangle - Topics

Class:
[ODShape](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

SetRegion (OS/2)

SetRegion (OS/2) - Syntax

This method replaces the geometric representation of this shape with the specified region.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODRgnHandle    rgn;

SetRegion(rgn);
```

SetRegion (OS/2) Parameter - rgn

rgn (ODRgnHandle) - input
A handle to the GPI region.

SetRegion (OS/2) - Return Value

None.

SetRegion (OS/2) - Parameters

rgn (ODRgnHandle) - input
A handle to the GPI region.

None.

SetRegion (OS/2) - Remarks

The region is consumed by the shape, and the caller must not use or destroy it. This call is identical to the following call:


```
s->SetPlatformShape(kODGPI, (ODPlatformShape)hrgn);
```

SetRegion (OS/2) - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

Subtract

Subtract - Syntax

This method modifies this shape by subtracting the specified shape from it.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *diffShape;
ODShape      *rv;

rv = Subtract(diffShape);
```

Subtract Parameter - diffShape

diffShape (ODShape *) - input

A reference to the shape to be subtracted from this shape.

Subtract Return Value - rv

rv (ODShape *) - returns

A reference to this shape after the subtraction operation.

Subtract - Parameters

diffShape (ODShape *) - input
A reference to the shape to be subtracted from this shape.

rv (ODShape *) - returns
A reference to this shape after the subtraction operation.

Subtract - Remarks

After this method executes successfully, this shape is equivalent to its previous shape minus the specified shape.

Subtract - Exception Handling

kODErrNoShapeGeometry

The geometry mode of this shape is kODNeedsGeometry, but the other shape has no geometric representation.

kODErrOutOfMemory

There is not enough memory to subtract the shape.

Subtract - Topics

Class:

ODShape

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

Transform

Transform - Syntax

This method modifies this shape by applying the specified transform to it.

```

#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODTransform      *transform;
ODShape          *rv;

rv = Transform(transform);

```

Transform Parameter - transform

transform (ODTransform *) - input
A reference to the transform to be applied to this shape.

Transform Return Value - rv

rv (ODShape *) - returns
A reference to this shape after it has been transformed.

Transform - Parameters

transform (ODTransform *) - input
A reference to the transform to be applied to this shape.

rv (ODShape *) - returns
A reference to this shape after it has been transformed.

Transform - Remarks

You can use this method to move a shape from one coordinate system to another. For example, a facet's clip shape is in the coordinate system of the frame. To get it into the coordinate system of the canvas, (which you need to do prior to obtaining a clipping region) you have to transform it by the facet's external transform.

Shapes without a geometric representation may not be transformable except by simple offsets.

Transform - Exception Handling

kODErrNoShapeGeometry

The specified shape does not have enough geometric information to be transformed in this way.

Transform - Topics

Class:

ODShape

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Exception Handling](#)

Union

Union - Syntax

This method modifies this shape to be the union of this shape and the specified shape.

```
#define INCL_ODSHAPE
#define INCL_ODAPI
#include <os2.h>

ODShape      *unionShape;
ODShape      *rv;

rv = Union(unionShape);
```

Union Parameter - unionShape

unionShape (ODShape *) - inputA reference to the shape to be unioned with this shape.

Union Return Value - rv

rv (ODShape *) - returnsA reference to this shape after the union operation.

Union - Parameters

unionShape (ODShape *) - input
A reference to the shape to be unioned with this shape.

rv (ODShape *) - returns
A reference to this shape after the union operation.

Union - Remarks

After this method executes successfully, this shape is equivalent to the union of its previous shape and the specified shape.

Union - Exception Handling

kODErrNoShapeGeometry	The geometry mode of this shape is kODNeedsGeometry, but the other shape has no geometric representation.
kODErrOutOfMemory	There is not enough memory to compute the union.

Union - Topics

Class:
ODShape

Select an item:

- [Syntax](#)
- [Parameters](#)
- [Returns](#)
- [Remarks](#)
- [Exception Handling](#)

WriteShape

WriteShape - Syntax

This method writes this shape to the specified storage unit.

```
#define INCL_ODSHAPE
```

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;

WriteShape(storageUnit);
```

WriteShape Parameter - storageUnit

storageUnit (ODStorageUnit *) - input
A reference to the storage unit where the shape data is to be written.

WriteShape - Return Value

None.

WriteShape - Parameters

storageUnit (ODStorageUnit *) - input
A reference to the storage unit where the shape data is to be written.

None.

WriteShape - Remarks

Before calling this method, you must focus the storage unit to the property where the shape data is to be written.

If the shape can be represented as a polygon, it is written as such to the value of type kODPolygon in the focus property, replacing any polygon that was previously stored in that value or creating the value if it does not already exist.

WriteShape - Exception Handling

kODErrOutOfMemory

There is not enough memory to write the shape's data to the storage unit.

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or a value.

WriteShape - Related Methods

Related Methods

- [ODShape::ReadShape](#)
-

WriteShape - Topics

Class:

[ODShape](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

ODStatusLineExtension

Class Definition File: STATUS.IDL

Class Hierarchy

```
SOMObject
  ODOObject
    ODBaseSession
      ODSession
        ODStatusLineExtension
```

Description

The status line is used to display a single line of text. The part which places text on the status line is responsible for removing it. A part can set the status line text, but it has no further control. The document shell is responsible for managing the status line window. The size and shape of the status line is determined by the document shell. If the status line text is set to empty, the status line is cleared.

Parts can get the status line extension from the session and thereby place text into the status line. A part should request status line focus before it sends text to the status line and relinquishes focus when it no longer needs the status line.

The status line extension is accessible to part handlers through the [ODSession](#) class. This class receives the string from the part and displays it by calling the handle received on initialization.

Note: A part does not need to explicitly use this class to display status text for their menu items if they use the menu bar object's [SetMenuItemStatusText](#) method. The document shell then displays the status text associated with the menu item for that part.

Methods

The methods defined by the ODStatusLineExtension class include:

- [InitStatusLineExtension](#)
- [SetStatusLineText](#)

Overridden Methods

There are currently no methods overridden by the ODStatusLineExtension class.

InitStatusLineExtension (OS/2)

InitStatusLineExtension (OS/2) - Syntax

This method is called by the document shell to initialize the status line to receive text.

```
#define INCL_ODSTATUSLINEEXTENSION
#define INCL_ODAPI
#include <os2.h>

ODSession      *owner;

InitStatusLineExtension(owner);
```

InitStatusLineExtension (OS/2) Parameter - owner

owner (ODSession *) - input
A reference to the current session object.

InitStatusLineExtension (OS/2) - Return Value

None.

InitStatusLineExtension (OS/2) - Parameters

owner (ODSession *) - input
A reference to the current session object.

None.

InitStatusLineExtension (OS/2) - Remarks

Parts do not call this method.

InitStatusLineExtension (OS/2) - Topics

Class:

ODStatusLineExtension

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)

SetStatusLineText (OS/2)

SetStatusLineText (OS/2) - Syntax

This method sets the text displayed in the status line.

```
#define INCL_ODSTATUSLINEEXTENSION
#define INCL_ODAPI
#include <os2.h>

string      str;
ODFrame     *reqFrame;
ODBoolean   rv;

rv = SetStatusLineText(str, reqFrame);
```

SetStatusLineText (OS/2) Parameter - str

str ([string](#)) - input

A string of text to be placed in the status line. If a null string is passed, the status line is cleared.

SetStatusLineText (OS/2) Parameter - reqFrame

reqFrame (ODFrame *) - input

A reference to the part's current frame.

SetStatusLineText (OS/2) Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating success.

kODTrue

The status line text was successfully set.

kODFalse

The status line text was not set.

SetStatusLineText (OS/2) - Parameters

str ([string](#)) - input

A string of text to be placed in the status line. If a null string is passed, the status line is cleared.

reqFrame (ODFrame *) - input

A reference to the part's current frame.

rv ([ODBoolean](#)) - returns

A flag indicating success.

kODTrue

The status line text was successfully set.

kODFalse

The status line text was not set.

SetStatusLineText (OS/2) - Topics

Class:

ODStatusLineExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODStorageSystem

Class Definition File: ODSOR.IDL

Class Hierarchy

SOMObject

ODObject

ODBaseStorageSystem

ODStorageSystem

Description

An object of the ODStorageSystem class provides the functionality of the OpenDoc storage system.

The OpenDoc storage system is a high-level persistent storage mechanism that enables multiple part editors to share a single document file effectively. When a document is opened, the session object creates a single storage-system object. All parts of that document share the storage-system object; you can obtain a reference to it by calling the session object's [GetStorageSystem](#) method.

The storage-system object creates and maintains a collection of container objects. Each container object can hold one or more document objects, each of which contains one or more draft objects. Each draft contains a number of storage unit objects, which hold streams of stored data.

Each container object has a container type and an identifier; these two characteristics together uniquely identify the container. OpenDoc supports both file containers and memory containers. File containers are persistent across sessions, whereas memory containers are

transitory.

For more information related to container objects, see the class description for [ODContainer](#).

Methods

The methods defined by the ODStorageSystem class include:

- [AcquireContainer](#)
- [CreateContainer](#)
- [CreatePlatformTypeList](#)
- [CreateTypeList](#)
- [GetSession](#)
- [NeedSpace](#)

Overridden Methods

There are currently no methods overridden by the ODStorageSystem class.

AcquireContainer

AcquireContainer - Syntax

This method is called by the document shell or container applications to retrieve a reference to the container object with the specified container type and identifier.

```
#define INCL_ODSTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>

ODContainerType    containerType;
ODContainerID      *id;
ODContainer        *rv;

rv = AcquireContainer(containerType, id);
```

AcquireContainer Parameter - containerType

containerType ([ODContainerType](#)) - input

The type of the container object. This parameter can be set to one of the following values:

- kODBentoFileContainer
The Bento file container (for documents) on this platform.
- kODBentoMemoryContainer
The Bento memory container (for drag-and-drop operations and the clipboard) on this platform.
- kODDefaultFileContainer
The default file container (for documents) on this platform.
- kODDefaultMemoryContainer
The default memory container (for drag-and-drop operations and the clipboard) on this platform.

AcquireContainer Parameter - id

id (ODContainerID *) - input
A container ID whose buffer contains data identifying the container object.

AcquireContainer Return Value - rv

rv (ODContainer *) - returns
A reference to the specified container object.

AcquireContainer - Parameters

containerType (ODContainerType) - input
The type of the container object. This parameter can be set to one of the following values:

kODBentoFileContainer	The Bento file container (for documents) on this platform.
kODBentoMemoryContainer	The Bento memory container (for drag-and-drop operations and the clipboard) on this platform.
kODDefaultFileContainer	The default file container (for documents) on this platform.
kODDefaultMemoryContainer	The default memory container (for drag-and-drop operations and the clipboard) on this platform.

id (ODContainerID *) - input
A container ID whose buffer contains data identifying the container object.

rv (ODContainer *) - returns
A reference to the specified container object.

AcquireContainer - Remarks

The document shell and container applications call this method when opening an OpenDoc container.

The structure of the data in the *id* parameter's buffer depends on the type of container, specified by the *containerType* parameter. For example, the identifier for a file container is a specification for a file-system file; the identifier for a memory container is a handle for a relocatable memory block.

When the structure passed as the *id* parameter is no longer needed, the caller should deallocate that structure and its buffer.

This method increments the reference count of the returned container object. When the caller has finished using that container object, it should call the container's [Release](#) method.

AcquireContainer - Exception Handling

kODErrCannotCreateContainer

The specified container type is not valid.

AcquireContainer - Related Methods

Related Methods

- [ODStorageSystem::CreateContainer](#)
-

AcquireContainer - Topics

Class:

[ODStorageSystem](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

CreateContainer

CreateContainer - Syntax

This method is called by the document shell or container part to create a container object with the specified container type and identifier.

```
#define INCL_ODTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>

ODContainerType    containerType;
ODContainerID      *id;
ODContainer        *rv;

rv = CreateContainer(containerType, id);
```

CreateContainer Parameter - containerType

containerType ([ODContainerType](#)) - input

The type of the container object. This parameter can be set to one of the following values:

kODBentoFileContainer

The Bento file container (for documents) on this platform.

kODBentoMemoryContainer
The Bento memory container (for drag-and-drop operations and the clipboard) on this platform.

kODDefaultFileContainer
The default file container (for documents) on this platform.

kODDefaultMemoryContainer
The default memory container (for drag-and-drop operations and the clipboard) on this platform.

CreateContainer Parameter - id

id (ODContainerID *) - input
A container ID whose buffer contains data identifying the container object.

CreateContainer Return Value - rv

rv (ODContainer *) - returns
A reference to the newly created container object.

CreateContainer - Parameters

containerType (ODContainerType) - input
The type of the container object. This parameter can be set to one of the following values:

kODBentoFileContainer
The Bento file container (for documents) on this platform.

kODBentoMemoryContainer
The Bento memory container (for drag-and-drop operations and the clipboard) on this platform.

kODDefaultFileContainer
The default file container (for documents) on this platform.

kODDefaultMemoryContainer
The default memory container (for drag-and-drop operations and the clipboard) on this platform.

id (ODContainerID *) - input
A container ID whose buffer contains data identifying the container object.

rv (ODContainer *) - returns
A reference to the newly created container object.

CreateContainer - Remarks

The structure of the data in the *id* parameter's buffer depends on the type of container, as specified by the *containerType* parameter. For example, the identifier for a file container is a specification for a file-system file; the identifier for a memory container is a handle for a relocatable memory block.

The physical container corresponding to the specified container type and container identifier must exist when this method is called.

When the structure passed as the *id* parameter is no longer needed, the caller should deallocate that structure and its buffer.

This method initializes the reference count of the returned container. When the caller has finished using that container, it should call the

container's [Release](#) method.

CreateContainer - Exception Handling

kODErrCannotCreateContainer

The specified container type is not valid.

kODErrContainerExists

A container already exists with the specified container type and container identifier.

CreateContainer - Related Methods

Related Methods

- [ODStorageSystem::AcquireContainer](#)
-

CreateContainer - Example Code

The following code fragment creates an embedded container.

```
typedef struct ODEmbeddedContainerID_struct {
    CMValue      cmValue;
    ODBoolean     shouldMerge;
} ODEmbeddedContainerID;

ODEmbeddedContainerID  containerID;
ODByteArray*          ba = CreateByteArray(&containerID,
                                           sizeof(ODEmbeddedContainerID));

container = (ODEmbeddedContainer*) fStorageSystem(ev)->CreateContainer(ev,
                              kODBentoEmbeddedContainer,
                              ba);

DisposeByteArray(ba);
```

The following code fragment creates an memory container.

```
Handle      handle = ODNewHandle(0);
ODByteArray* ba;

ODLockHandle(handle);
ba = CreateByteArray(&handle, sizeof(ODHandle));
ODUnlockHandle(handle);

container = (ODMemoryContainer*) fStorageSystem(ev)->CreateContainer(ev,
                              kODBentoMemoryContainer,
                              ba);

DisposeByteArray(ba);
```

The following code fragment creates an file container.

```
ODFileSpec  fileSpec;
PlatformFile* file = new PlatformFile;
```

```

strcpy(fileSpec.name, "TEST.ODP" );
file->Specify( &fileSpec );
file->Create(kOpenDocShellSignature, kOpenDocShellSignature, 0);

ODByteArray* ba = CreateByteArray(fsSpec, sizeof(ODFileSpec));

container = (ODFileContainer*) fStorageSystem(ev)->CreateContainer(ev,
                                                                kODBentoFileContainer,
                                                                ba);

DisposeByteArray(ba);

```

CreateContainer - Topics

Class:

ODStorageSystem

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Example Code](#)

[Related Methods](#)

CreatePlatformTypeList

CreatePlatformTypeList - Syntax

This method creates or copies a platform type list.

```

#define INCL_ODSTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>

ODPlatformTypeList    *typeList;
ODPlatformTypeList    *rv;

rv = CreatePlatformTypeList(typeList);

```

CreatePlatformTypeList Parameter - typeList

typeList (ODPlatformTypeList *) - input

A reference to the platform type list to be duplicated, or kODNULL to create an empty platform type list.

CreatePlatformTypeList Return Value - rv

rv (ODPlatformTypeList *) - returns
A reference to the newly created platform type list.

CreatePlatformTypeList - Parameters

typeList (ODPlatformTypeList *) - input
A reference to the platform type list to be duplicated, or KODNULL to create an empty platform type list.

rv (ODPlatformTypeList *) - returns
A reference to the newly created platform type list.

CreatePlatformTypeList - Remarks

Your part calls this method.

CreatePlatformTypeList - Topics

Class:
ODStorageSystem

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

CreateTypeList

CreateTypeList - Syntax

This method creates or copies a type list.

```
#define INCL_ODSTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeList      *typeList;
```

```
ODTypeList      *rv;  
  
rv = CreateTypeList (typeList);
```

CreateTypeList Parameter - typeList

typeList (ODTypeList *) - input
A reference to the type list to be duplicated, or kODNULL to create an empty type list.

CreateTypeList Return Value - rv

rv (ODTypeList *) - returns
A reference to the newly created type list.

CreateTypeList - Parameters

typeList (ODTypeList *) - input
A reference to the type list to be duplicated, or kODNULL to create an empty type list.

rv (ODTypeList *) - returns
A reference to the newly created type list.

CreateTypeList - Remarks

You can call this method if your part needs a list of types (for example, a list of part kinds it supports).

If the *typeList* parameter is a reference to an existing type list, the new type list is initialized to contain a copy of each element in the existing type list. The elements in the new list are in the same order as the elements of the existing type list. Because each element of the existing list is a pointer to an ISO string, the pointers themselves are not added to the new type list; instead, each string is copied and a pointer to the new copy is added to the type list.

CreateTypeList - Topics

Class:
ODStorageSystem

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetSession

GetSession - Syntax

This method returns a reference to the current session object.

```
#define INCL_ODSTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>

ODSession      *rv;

rv = GetSession();
```

GetSession Return Value - rv

rv (ODSession *) - returns
A reference to the session object which created this storage system.

GetSession - Parameters

rv (ODSession *) - returns
A reference to the session object which created this storage system.

GetSession - Remarks

Your part typically calls its storage unit's [GetSession](#) method instead of this method.

GetSession - Related Methods

Related Methods

- [ODStorageUnit::GetSession](#)

GetSession - Topics

Class:
ODStorageSystem

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

NeedSpace

NeedSpace - Syntax

This method notifies this storage system to reserve memory for future use.

```
#define INCL_ODSTORAGESYSTEM
#define INCL_ODAPI
#include <os2.h>

ODSize      memSize;
ODBoolean    doPurge;

NeedSpace(memSize, doPurge);
```

NeedSpace Parameter - memSize

memSize ([ODSize](#)) - input
The size of the desired memory block.

NeedSpace Parameter - doPurge

doPurge ([ODBoolean](#)) - input
A flag indicating whether this operation should trigger the [Purge](#) method to obtain the requested memory.

kODTrue	
kODFalse	This object should purge memory to obtain the requested memory.
	This object should not purge memory.

NeedSpace - Return Value

None.

NeedSpace - Parameters

memSize ([ODSize](#)) - input

The size of the desired memory block.

doPurge ([ODBoolean](#)) - input

A flag indicating whether this operation should trigger the [Purge](#) method to obtain the requested memory.

kODTrue

This object should purge memory to obtain the requested memory.

kODFalse

This object should not purge memory.

None.

NeedSpace - Remarks

Your part can call this method when it anticipates the need for a large memory block. This method is not guaranteed to generate the memory requested and should be used with caution as it may be a slow operation.

If memory cannot be allocated and the *doPurge* parameter is kODTrue, then this method calls the [Purge](#) method associated with this storage system's container objects (and transitively its document objects, draft objects, and storage-unit objects).

NeedSpace - Topics

Class:

ODStorageSystem

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

ODStorageUnit

Class Definition File: STORAGEU.IDL

Class Hierarchy

SOMObject
ODObject
ODRefCntObject
ODStorageUnit

Description

An object of the ODStorageUnit class represents the basic unit of persistent storage.

The ODStorageUnit class is implemented differently for different platforms and storage mechanisms.

The set of related classes, [ODContainer](#), [ODDocument](#), [ODDraft](#), and [ODStorageUnit](#), is called a *container suite*. Container suite classes are implemented as an integrated set for each platform and storage mechanism because they work intimately with one another at many levels.

Every persistent object has an associated storage unit where it stores its data persistently. Storage units are also used for data-transfer operations; the clipboard and the drag-and-drop object each have a content storage unit where they store the data to be transferred. Because storage units may no longer be valid as soon as a data transfer is complete, you should never cache a reference to a data-transfer object's storage unit.

Your part creates a new storage unit by calling its draft's [CreateStorageUnit](#) method; it accesses an existing storage unit by calling its draft's [AcquireStorageUnit](#) method.

A storage unit consists of one or more *properties*, each of which has a unique name within the storage unit and is distinguished by the kind of information it contains (such as "name" or "content"). For example, the properties in your part's storage unit are used to store persistently both the content of your part and also supplemental information. OpenDoc defines a standard set of properties for all parts; you can define additional properties for information specific to your particular part. A property consists of one or more data streams, called *values*, each of which has a named type. Each property can have only one value of a given type.

In a data-transfer operation, the source part writes data to one of more values of the data-transfer object's content storage unit; the destination part reads from those values. The source part can write either the data to be transferred or a *promise*, which is a specification of data to be transferred at a future time. A value that contains promises data is called a promise value; a value that does not contain promise data is called a regular value. When and if a destination tries to read the promise, the storage unit first *resolves* the promise by asking the source part to fulfill it. The source part *fulfills* the promise by replacing the promise in the storage unit with the actual data that was promised.

OpenDoc assigns a run-time identifier, a *storage-unit ID*, to each of its storage units. A storage-unit ID is a nonpersistent identifier for a storage unit that is unique within its draft (storage-unit IDs are not unique across drafts and do not persist across sessions). You can use the storage-unit ID to identify storage units and to compare two storage units for equality. The ID of the storage unit for a persistent object is also used as the ID of the object itself. If you retain the ID of a persistent object when you release it, you can use that ID to recreate the object from the data stored in its storage unit. For example, when a frame is scrolled out of view, you can save its ID and release it; if the frame is later scrolled back into view, you can obtain a reference to it by passing the saved ID to the draft's [AcquireFrame](#) method. If the frame has been purged since you released it, the [AcquireFrame](#) method recreates it from its stored data.

Storage units can maintain persistent references to other storage units in the same draft. A *persistent reference* stored in a storage unit value is an opaque type that identifies another storage unit in the same draft. Whereas the ID of a storage unit identifies another storage unit within the current session, a persistent reference to the storage unit identifies it persistently across sessions. Persistent references permit complex run-time relationships between objects to be stored externally and later reconstructed; for example, the embedding relationships of the parts within a draft are preserved by persistent references to the parts's storage unit. Persistent references can be either strong or weak. In a clone operation, copies are made of all storage units referenced by strong persistent reference in the object being cloned. A weak persistent reference is typically ignored during cloning; however, if a storage unit is cloned because there are strong persistent references to it, then weak persistent references to the storage unit are preserved.

OpenDoc allows you to *focus* a storage unit on the particular data of interest, called the *focus context*. Before reading or writing to a storage unit, for example, you must focus on the data stream defined by a particular value of a particular property. At any time, the storage unit can be in one of the following states:

- Unfocused. When a storage unit is unfocused, its focus context is undefined.
- Focused on the entire storage unit. When the focus context is the storage unit, the data of interest includes all properties and all their values.
- Focused on a particular property. When the focus context is a property, the data of interest includes all values of the focused property.
- Focused on a particular value of a particular property. When the focus context is a value, the data of interest is the data stream corresponding to a focused value.

To define a focus context, you can specify a property by its name or a position code. Within a given property, you can specify the value of interest by a value type, a position code, or a value index. A position code is a constant that specifies either an absolute position for the property or value, or a position relative to the property or value in the current focus context. Position codes allow you to access the next or previous property within a set of properties in a storage unit or the next or previous value within a set of values in the same property. A *value index* is a number representing the ordinal position of the value within the property. The first value created for a property has index 1; the second, 2; and so on.

When the storage unit is focused on a value of a property, you can read data from and write data to the corresponding data stream. The storage unit has a zero-based offset that specifies the current read/write position in the stream. When the storage unit is first focused, the

offset is 0, indicating the beginning of the stream. Each read and write advances the offset by the number of bytes that were read or written. The storage unit also has methods that allow you to get and set the current offset.

You can call methods of a storage unit to create related objects that make it easy to work with the data in the storage unit.

- A storage-unit view represents the storage unit prefocused on a particular focus context. You can pass a storage-unit view among your software components to give them access to the particular focused data stream.
- A storage-unit cursor represents a focus context. You can create storage-unit cursors for focus contexts that you access frequently, then use those cursors to switch the focus from one context to another.
- A storage-unit reference iterator allows you to access all persistent references in the currently focused value of the storage unit.

For more information about these objects, see the descriptions of the classes [ODStorageUnitView](#), [ODStorageUnitCursor](#), and [ODStorageUnitRefIterator](#).

Methods

The methods defined by the `ODStorageUnit` class include:

Accessing and Storing

- [CloneInto](#)
- [Externalize](#)
- [Internalize](#)
- [Lock](#)
- [Unlock](#)

Manipulating Storage Units

- [AddProperty](#)
- [AddValue](#)
- [Remove](#)
- [CountProperties](#)
- [CountValues](#)
- [GetID](#)
- [GetName](#)
- [GetSize](#)
- [SetName](#)

Manipulating Foci

- [Exists](#)
- [ExistsWithCursor](#)
- [Focus](#)
- [FocusWithCursor](#)
- [GetProperty](#)

Manipulating Values

- [DeleteValue](#)
- [GetGenerationNumber](#)
- [GetOffset](#)
- [GetType](#)
- [GetValue](#)
- [IncrementGenerationNumber](#)
- [InsertValue](#)
- [SetOffset](#)
- [SetType](#)
- [SetValue](#)

Manipulating Persistent References

- [GetIDFromStorageUnitRef](#)
- [GetStrongStorageUnitRef](#)
- [GetWeakStorageUnitRef](#)
- [IsStrongStorageUnitRef](#)
- [IsValidStorageUnitRef](#)
- [IsWeakStorageUnitRef](#)
- [RemoveStorageUnitRef](#)
- [SetStorageUnitRef](#)

Manipulating Promises

- [ClearAllPromises](#)
- [GetPromiseValue](#)

- [IsPromiseValue](#)
- [ResolveAllPromises](#)
- [SetPromiseValue](#)

Creating Objects

- [CreateCursor](#)
- [CreateCursorWithFocus](#)
- [CreateStorageUnitRefIterator](#)
- [CreateView](#)

Utility Routines

- [GetDraft](#)
- [GetSession](#)

Overridden Methods

There are currently no methods overridden by the ODStorageUnit class.

AddProperty

AddProperty - Syntax

This method adds a property with the specified name to this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODPropertyName    propertyName;
ODStorageUnit     *rc;

rc = AddProperty (propertyName);
```

AddProperty Parameter - propertyName

propertyName ([ODPropertyName](#)) - input
The name of the property to be added.

AddProperty Return Value - rc

rc ([ODStorageUnit *](#)) - returns
A reference to this storage unit.

AddProperty - Parameters

propertyName ([ODPropertyName](#)) - input
The name of the property to be added.

rc ([ODStorageUnit *](#)) - returns
A reference to this storage unit.

AddProperty - Remarks

If the storage unit does not already contain a property with the specified name, the new property is added and this storage unit is focused on the newly added property. Otherwise, this storage unit is focused on the existing property with the specified name.

AddProperty - Exception Handling

[kODErrCannotAddProperty](#)
[kODErrIllegalNullPropertyInput](#)
[kODErrZeroRefCount](#)

Cannot add the specified property to this storage unit.
The specified property name is null.
This storage unit has a reference count of 0.

AddProperty - Related Methods

Related Methods

- [ODStorageUnit::GetProperty](#)
 - [ODStorageUnit::Remove](#)
-

AddProperty - Topics

Class:
[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

AddValue

AddValue - Syntax

This method adds a value of the specified type to the currently focused property.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType      type;
ODStorageUnit    *rc;

rc = AddValue(type);
```

AddValue Parameter - type

type (ODValueType) - input
The type of value to be added.

AddValue Return Value - rc

rc (ODStorageUnit *) - returns
A reference to this storage unit.

AddValue - Parameters

type (ODValueType) - input
The type of value to be added.

rc (ODStorageUnit *) - returns
A reference to this storage unit.

AddValue - Remarks

If the focused property does not already contain a value of the specified type, the new value is added and this storage unit is focused on the newly added value. Otherwise, this storage unit is focused on the existing value with the specified value type.

AddValue - Exception Handling

kODErrInvalidType
kODErrCannotAddType
kODErrUnfocusedStorageUnit
kODErrZeroRefCount

The specified value type is null.
The specified value type is improperly formed or illegal.
This storage unit is not focused on a property.
This storage unit has a reference count of 0.

AddValue - Related Methods

Related Methods

- [ODStorageUnit::Remove](#)

AddValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

ClearAllPromises

ClearAllPromises - Syntax

This method changes all promise values in this storage unit into regular values.

```
#define INCL_ODSTORAGEUNIT  
#define INCL_ODAPI  
#include <os2.h>
```

```
ClearAllPromises();
```

ClearAllPromises - Return Value

None.

ClearAllPromises - Parameters

None.

ClearAllPromises - Remarks

This method does not change the data in any of the promise values; it simply changes the values from promise values to regular values.

After this method executes successfully, the storage unit is unfocused.

ClearAllPromises - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

ClearAllPromises - Related Methods

Related Methods

- [ODStorageUnit::ResolveAllPromises](#)
-

ClearAllPromises - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CloneInto

CloneInto - Syntax

This method copies all properties and values of this storage unit to the specified destination storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODDraftKey      key;
ODStorageUnit   *destStorageUnit;
ODID            scopeID;

CloneInto(key, destStorageUnit, scopeID);
```

CloneInto Parameter - key

key (ODDraftKey) - input
The draft key identifying this cloning operation.

CloneInto Parameter - destStorageUnit

destStorageUnit (ODStorageUnit *) - input
A reference to the destination storage unit to which the data is to be copied.

CloneInto Parameter - scopeID

scopeID (ODID) - input
The ID of the frame that defines the scope of this cloning operation or kODIDAll if all references objects are within scope.

CloneInto - Return Value

None.

CloneInto - Parameters

key ([ODDraftKey](#)) - input

The draft key identifying this cloning operation.

destStorageUnit ([ODStorageUnit *](#)) - input

A reference to the destination storage unit to which the data is to be copied.

scopeID ([ODID](#)) - input

The ID of the frame that defines the scope of this cloning operation or kODIDAll if all references objects are within scope.

None.

CloneInto - Remarks

This method is not called by parts. Your part should call its draft's [Clone](#) or [WeakClone](#) method instead of this method.

If this storage unit has persistent references to other objects, the *scopeID* parameter determines which of the referenced objects are within the scope of this cloning operation. Typically, the *scopeID* parameter is the ID of a frame, and only those objects embedded in that frame are within scope.

This method copies this storage unit's data into the specified destination storage unit. If this storage unit has persistent references to other objects, this method clones any persistent referenced objects that are within the scope of this cloning operation. Objects referenced by strong persistent references are strongly cloned by recursive calls to the [Clone](#) method; objects referenced by weak persistent references are weakly cloned by calls to the [WeakClone](#) method.

CloneInto - Exception Handling

kODErrInvalidDraftKey
kODErrZeroRefCount

The specified draft key is invalid.
This storage unit has a reference count of 0.

CloneInto - Related Methods

Related Methods

- [ODDraft::Clone](#)
 - [ODDraft::WeakClone](#)
-

CloneInto - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CountProperties

CountProperties - Syntax

This method returns the number of properties in this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong    rc;

rc = CountProperties();
```

CountProperties Return Value - rc

rc ([ODULong](#)) - returns
The number of properties in this storage unit.

CountProperties - Parameters

rc ([ODULong](#)) - returns
The number of properties in this storage unit.

CountProperties - Remarks

After this method executes successfully, the focus context of this storage unit is not changed.

CountProperties - Exception Handling

CountProperties - Related Methods

Related Methods

- [ODStorageUnit::CountValues](#)
-

CountProperties - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

CountValues

CountValues - Syntax

This method returns the number of values in the current focus context for this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rc;
```

```
rc = CountValues();
```

CountValues Return Value - rc

rc ([ODULong](#)) - returns

The number of values in this storage unit.

CountValues - Parameters

rc ([ODULong](#)) - returns
The number of values in this storage unit.

CountValues - Remarks

This storage unit must be focused on property or a value, not the entire storage unit.

CountValues - Exception Handling

<code>kODErrUnfocusedStorageUnit</code>	This storage unit is not focused on a property or a value.
<code>kODErrZeroRefCount</code>	This storage unit has a reference count of 0.

CountValues - Related Methods

Related Methods

- [ODStorageUnit::CountProperties](#)
-

CountValues - Topics

Class:
`ODStorageUnit`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CreateCursor

CreateCursor - Syntax

This method creates a storage-unit cursor representing the specified focus context for this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODPropertyName      propertyName;
ODValueType          valueType;
ODValueIndex         valueIndex;
ODStorageUnitCursor *rc;

rc = CreateCursor(propertyName, valueType,
                  valueIndex);
```

CreateCursor Parameter - propertyName

propertyName ([ODPropertyName](#)) - input

The name of the property in the focus context or KODNULL if the focus context is the entire storage unit.

CreateCursor Parameter - valueType

valueType ([ODValueType](#)) - input

The value type of the value in the focus context, KODTypeAll for all value types, or KODNULL to ignore this parameter.

CreateCursor Parameter - valueIndex

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the focus context, KODIndexAll for all value indexes, or 0 to ignore this parameter.

CreateCursor Return Value - rc

rc ([ODStorageUnitCursor *](#)) - returns

A reference to the newly created storage-unit cursor.

CreateCursor - Parameters

propertyName ([ODPropertyName](#)) - input

The name of the property in the focus context or KODNULL if the focus context is the entire storage unit.

valueType ([ODValueType](#)) - input

The value type of the value in the focus context, KODTypeAll for all value types, or KODNULL to ignore this parameter.

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the focus context, KODIndexAll for all value indexes, or 0 to ignore this parameter.

rc ([ODStorageUnitCursor *](#)) - returns

A reference to the newly created storage-unit cursor.

CreateCursor - Remarks

Your part calls this method; its parameters specify the focus context for the storage-unit cursor.

- To specify the entire storage unit as the focus context, pass KODNULL in the *propertyName* parameter, KODTypeAll in the *valueType* parameter, and KODIndexAll in the *valueIndex* parameter.
- To specify a property as the focus context, pass its name in the *propertyName* parameter, KODTypeAll in the *valueType* parameter, and KODIndexAll in the *valueIndex* parameter.
- To specify a value as the focus context, pass the name of the property containing the value in the *propertyName* parameter. You can specify the value by either its type or its index:
 - To use its type, pass the type of the desired value in the *valueType* parameter and 0 in the *valueIndex* parameter.
 - To use its index, pass KODNULL in the *valueType* parameter and the index of the desired value in the *valueIndex* parameter.

CreateCursor - Exception Handling

KODErrZeroRefCount

This storage unit has a reference count of 0.

CreateCursor - Related Methods

Related Methods

- [ODStorageUnit::CreateCursorWithFocus](#)

CreateCursor - Topics

Class:

ODStorageUnit

Select an item:

CreateCursorWithFocus

CreateCursorWithFocus - Syntax

This method creates a storage-unit cursor representing the current focus context of this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitCursor    *rc;

rc = CreateCursorWithFocus();
```

CreateCursorWithFocus Return Value - rc

rc (ODStorageUnitCursor *) - returns
A reference to the newly created storage-unit cursor.

CreateCursorWithFocus - Parameters

rc (ODStorageUnitCursor *) - returns
A reference to the newly created storage-unit cursor.

CreateCursorWithFocus - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

CreateCursorWithFocus - Related Methods

Related Methods

- [ODStorageUnit::CreateCursor](#)

CreateCursorWithFocus - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

CreateStorageUnitRefIterator

CreateStorageUnitRefIterator - Syntax

This method creates a storage-unit reference iterator for the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRefIterator    *rc;

rc = CreateStorageUnitRefIterator();
```

CreateStorageUnitRefIterator Return Value - rc

rc (ODStorageUnitRefIterator *) - returns

A reference to a storage-unit reference iterator used to traverse all persistent references in the currently focused value.

CreateStorageUnitRefIterator - Parameters

rc (ODStorageUnitRefIterator *) - returns

A reference to a storage-unit reference iterator used to traverse all persistent references in the currently focused value.

CreateStorageUnitRefIterator - Remarks

You can call this method if you need to perform an operation on all storage-unit references in the currently focused value.

While you are using the returned storage-unit reference iterator, you must not modify the focused value; in particular, you must not call any of the following methods of this storage unit: [DeleteValue](#), [Internalize](#), [Remove](#), [RemoveStorageUnitRef](#), [SetStorageUnitRef](#), [SetType](#), or [SetValue](#). Furthermore, you must not delete this storage unit.

You must delete the returned storage-unit reference iterator when it is no longer needed.

CreateStorageUnitRefIterator - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

CreateStorageUnitRefIterator - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

CreateView

CreateView - Syntax

This method creates a storage-unit view for this storage unit with its current focus context.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *rc;

rc = CreateView();
```

CreateView Return Value - rc

rc (ODStorageUnitView *) - returns
A reference to the newly created storage-unit view.

CreateView - Parameters

rc (ODStorageUnitView *) - returns
A reference to the newly created storage-unit view.

CreateView - Remarks

Your part calls this method.

CreateView - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

CreateView - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

DeleteValue

DeleteValue - Syntax

This method deletes data from the currently focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong    length;

DeleteValue(length);
```

DeleteValue Parameter - length

length ([ODULong](#)) - input
The number of bytes of data to be deleted.

DeleteValue - Return Value

None.

DeleteValue - Parameters

length ([ODULong](#)) - input
The number of bytes of data to be deleted.

None.

DeleteValue - Remarks

You call this method to delete data from the currently focused value. If that value is a promise value, the promise is fulfilled before the data is deleted. This method starts deleting data at the current offset and stops after deleting the number of bytes specified by the *length* parameter or after reaching the end of the data in the currently focused value, whichever comes first.

DeleteValue - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

DeleteValue - Related Methods

Related Methods

- [ODStorageUnit::InsertValue](#)
- [ODStorageUnit::Remove](#)

DeleteValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

Exists

Exists - Syntax

This method indicates whether the specified focus context exists in this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODPropertyName    propertyName;
ODValueType        valueType;
ODValueIndex       valueIndex;
ODBoolean          rc;
```

```
rc = Exists(propertyName, valueType, valueIndex);
```

Exists Parameter - propertyName

propertyName ([ODPropertyName](#)) - input
The name of the property in the focus context or kODNULL for the currently focused property.

Exists Parameter - valueType

valueType ([ODValueType](#)) - input
The value type of the value in the focus context, kODTypeAll for all value types, or kODNULL to ignore this parameter.

Exists Parameter - valueIndex

valueIndex ([ODValueIndex](#)) - input
The value index of the value in the focus context, kODIndexAll for all value indexes, or 0 to ignore this parameter.

Exists Return Value - rc

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified focus context exists in this storage unit.

kODTrue	The specified focus exists this storage unit.
kODFalse	The specified focus does not exist this storage unit.

Exists - Parameters

propertyName ([ODPropertyName](#)) - input
The name of the property in the focus context or kODNULL for the currently focused property.

valueType ([ODValueType](#)) - input
The value type of the value in the focus context, kODTypeAll for all value types, or kODNULL to ignore this parameter.

valueIndex ([ODValueIndex](#)) - input
The value index of the value in the focus context, kODIndexAll for all value indexes, or 0 to ignore this parameter.

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified focus context exists in this storage unit.

kODTrue	The specified focus exists this storage unit.
kODFalse	The specified focus does not exist this storage unit.

Exists - Remarks

You can call this method to see whether you can focus this storage unit on the specified focus context; however, this method does not change the current focus context. The parameters specify the context to be checked.

- To specify a property as the focus context, pass its name in the *propertyName* parameter or `KODNULL` for the currently focused property. Pass `KODTypeAll` in the *valueType* parameter and `KODIndexAll` in the *valueIndex* parameter.
- To specify a value as the focus context, pass the name of the property containing the value in the *propertyName* parameter or `KODNULL` for the currently focused property. You can specify the value by either its type or its index:
 - To use its type, pass the type of the desired value in the *valueType* parameter and 0 in the *valueIndex* parameter.
 - To use its index, pass `KODNULL` in the *valueType* parameter and the index of the desired value in the *valueIndex* parameter.

If this method returns `KODTrue`, it is safe to call the [Focus](#) method with the specified focus context.

Exists - Exception Handling

`KODErrInvalidProperty`
`KODErrInvalidType`
`KODErrZeroRefCount`

The specified property name is improperly formed or illegal.
The specified value type is improperly formed or illegal.
This storage unit has a reference count of 0.

Exists - Related Methods

Related Methods

- [ODStorageUnit::ExistsWithCursor](#)
- [ODStorageUnit::Focus](#)

Exists - Topics

Class:

`ODStorageUnit`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

ExistsWithCursor

ExistsWithCursor - Syntax

This method indicates whether the focus context represented by the specified storage-unit cursor exists in this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitCursor    *cursor;
ODBoolean              rc;

rc = ExistsWithCursor(cursor);
```

ExistsWithCursor Parameter - cursor

cursor (ODStorageUnitCursor *) - input

A reference to the storage-unit cursor representing the focus context to be tested.

ExistsWithCursor Return Value - rc

rc (ODBoolean) - returns

A flag indicating whether the focus context specified by the storage-unit cursor exists in this storage unit

kODTrue

The focus context specified by the storage-unit cursor exists in this storage unit

kODFalse

The focus context specified by the storage-unit cursor does not exist in this storage unit

ExistsWithCursor - Parameters

cursor (ODStorageUnitCursor *) - input

A reference to the storage-unit cursor representing the focus context to be tested.

rc (ODBoolean) - returns

A flag indicating whether the focus context specified by the storage-unit cursor exists in this storage unit

kODTrue

The focus context specified by the storage-unit cursor exists in this storage unit

kODFalse

The focus context specified by the storage-unit cursor does not exist in this storage unit

ExistsWithCursor - Remarks

You can call this method to see whether you can focus this storage unit using the specified storage-unit cursor; however, this method does not change the current focus context.

If this method returns `kODTrue`, it is safe to call the [FocusWithCursor](#) method with the specified storage-unit cursor.

ExistsWithCursor - Exception Handling

`kODErrZeroRefCount`

This storage unit has a reference count of 0.

ExistsWithCursor - Related Methods

Related Methods

- [ODStorageUnit::Exists](#)
- [ODStorageUnit::FocusWithCursor](#)

ExistsWithCursor - Topics

Class:

`ODStorageUnit`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

Externalize

Externalize - Syntax

This method resolves all promises in this storage unit and saves all properties and values to persistent, external storage.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnit    *rc;
```

```
rc = Externalize();
```

Externalize Return Value - rc

rc (ODStorageUnit *) - returns
A reference to this storage unit.

Externalize - Parameters

rc (ODStorageUnit *) - returns
A reference to this storage unit.

Externalize - Exception Handling

kODErrZeroRefCount This storage unit has a reference count of 0.

Externalize - Related Methods

Related Methods

- [ODStorageUnit::Internalize](#)
-

Externalize - Topics

Class:
ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Exception Handling](#)
[Related Methods](#)

Focus

Focus - Syntax

This method focuses this storage unit on the specified focus context.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODPropertyName      propertyName;
ODPositionCode      propertyPosCode;
ODValueType          valueType;
ODValueIndex         valueIndex;
ODPositionCode      valuePosCode;
ODStorageUnit        *rc;

rc = Focus(propertyName, propertyPosCode,
            valueType, valueIndex, valuePosCode);
```

Focus Parameter - propertyName

propertyName ([ODPropertyName](#)) - input

The name of the property in desired the focus context or KODNULL to ignore this parameter.

Focus Parameter - propertyPosCode

propertyPosCode ([ODPositionCode](#)) - input

The position code, relative to the desired focus context, of the property in the focus context or KODPosUndefined to ignore this parameter.

Focus Parameter - valueType

valueType ([ODValueType](#)) - input

The value type of the value in the desired focus context, KODTypeAll for all value types, or KODNULL to ignore this parameter.

Focus Parameter - valueIndex

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the desired focus context, KODIndexAll for all value indexes, or 0 to ignore this parameter.

Focus Parameter - valuePosCode

valuePosCode ([ODPositionCode](#)) - input

The position code, relative to the current focus context, of the value in the desired focus context, or kODPosUndefined to ignore this parameter.

Focus Return Value - rc

rc (ODStorageUnit *) - returns

A reference to this storage unit, focused on the specified focus context.

Focus - Parameters

propertyName ([ODPropertyName](#)) - input

The name of the property in desired the focus context or kODNULL to ignore this parameter.

propertyPosCode ([ODPositionCode](#)) - input

The position code, relative to the desired focus context, of the property in the focus context or kODPosUndefined to ignore this parameter.

valueType ([ODValueType](#)) - input

The value type of the value in the desired focus context, kODTypeAll for all value types, or kODNULL to ignore this parameter.

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the desired focus context, kODIndexAll for all value indexes, or 0 to ignore this parameter.

valuePosCode ([ODPositionCode](#)) - input

The position code, relative to the current focus context, of the value in the desired focus context, or kODPosUndefined to ignore this parameter.

rc (ODStorageUnit *) - returns

A reference to this storage unit, focused on the specified focus context.

Focus - Remarks

Your part calls this method; its parameters specify the desired focus context.

- To focus on the entire storage unit, pass kODNULL in the *propertyName* parameter, kODPosAll in the *propertyPosCode* parameter, kODTypeAll in the *valueType* parameter, 0 in the *valueIndex* parameter, and kODPosUndefined in the *valuePosCode* parameter.
- To focus on a property, either pass its name in the *propertyName* parameter, or pass kODNULL in the *propertyName* parameter and the appropriate code in the *propertyPosCode* parameter. Pass kODTypeAll in the *valueType* parameter, 0 in the *valueIndex* parameter, and kODPosUndefined in the *valuePosCode* parameter.
- To focus on a value, specify the property containing the value using either the *propertyName* parameter or the *propertyPosCode* parameter, as described in the previous item. You can specify the value by its type, its index, or its position.

- To use its type, pass the type of the desired value in the *valueType* parameter. Pass 0 in the *valueIndex* parameter and kODPosUndefined in the *valuePosCode* parameter.
- To use its index, pass kODNULL in the *valueType* parameter, the index of the desired value in the *valueIndex* parameter, and kODPosUndefined in the *valuePosCode* parameter.
- To use its position code, pass kODNULL as the *valueType* parameter, 0 as the *valueIndex* parameter, and the appropriate code as the *valuePosCode* parameter.

After this method executes successfully, this storage unit's offset is 0.

Before calling this method, you can call the [Exists](#) method to check whether the specified focus context exists.

Focus - Exception Handling

kODInvalidProperty

The specified property name is improperly formed or illegal.

kODErrInvalidType

The value type parameter is unknown or null.

kODErrInvalidValue

This storage unit does not contain the specified value.

kODErrUnsupportedPosCode

One of the specified position codes is not supported.

kODErrValueIndexOutOfRange

The specified property has no value at the specified index.

kODErrZeroRefCount

This storage unit has a reference count of 0.

Focus - Related Methods

Related Methods

- [ODStorageUnit::Exists](#)
- [ODStorageUnit::FocusWithCursor](#)

Focus - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

FocusWithCursor

FocusWithCursor - Syntax

This method focuses this storage unit on the focus context represented by the specified storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitCursor    *cursor;
ODStorageUnit          *rc;

rc = FocusWithCursor(cursor);
```

FocusWithCursor Parameter - cursor

cursor (ODStorageUnitCursor *) - input
A reference to the storage-unit cursor representing the desired focus context.

FocusWithCursor Return Value - rc

rc (ODStorageUnit *) - returns
A reference to this storage unit, focused on the specified focus context.

FocusWithCursor - Parameters

cursor (ODStorageUnitCursor *) - input
A reference to the storage-unit cursor representing the desired focus context.

rc (ODStorageUnit *) - returns
A reference to this storage unit, focused on the specified focus context.

FocusWithCursor - Remarks

After this method executes successfully, this storage unit's offset is 0.

Before calling this method, you can call the [ExistsWithCursor](#) method to check whether the specified focus context exists.

FocusWithCursor - Exception Handling

kODerrIllegalNullSUCursorInput
kODerrInvalidProperty
kODerrInvalidValue

kODerrValueIndexOutOfRange

kODerrZeroRefCount

The *cursor* parameter is null.
The *cursor* parameter specifies a property that does not exist.
The *cursor* parameter specifies a value type that does not exist for the specified property.
The *cursor* parameter specifies a value index that is out of range for the specified property.
This storage unit has a reference count of 0.

FocusWithCursor - Related Methods

Related Methods

- [ODStorageUnit::ExistsWithCursor](#)
- [ODStorageUnit::Focus](#)

FocusWithCursor - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetDraft

GetDraft - Syntax

This method returns a reference to the draft object which created this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *rc;

rc = GetDraft();
```

GetDraft Return Value - rc

rc (ODDraft *) - returns

A reference to the draft object which created this storage unit.

GetDraft - Parameters

rc (ODDraft *) - returns

A reference to the draft object which created this storage unit.

GetDraft - Remarks

This method does not increment the reference count of the returned draft. For that reason, if you cache the returned draft, you should call its [Acquire](#) method to increment its reference count and then call its [Release](#) method when you are finished using it.

GetDraft - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetDraft - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

GetGenerationNumber

GetGenerationNumber - Syntax

This method returns the generation number of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong    rc;

rc = GetGenerationNumber();
```

GetGenerationNumber Return Value - rc

rc ([ODULong](#)) - returns
The generation number of the currently focused value, expressed as an unsigned 32-bit value.

GetGenerationNumber - Parameters

rc ([ODULong](#)) - returns
The generation number of the currently focused value, expressed as an unsigned 32-bit value.

GetGenerationNumber - Remarks

You can use the generation number of a value to tell whether the data in the value has changed. For example, your part could compare the number returned by this method with a saved generation number.

GetGenerationNumber - Exception Handling

<code>kODErrUnfocusedStorageUnit</code>	This storage unit is not focused on a value.
<code>kODErrZeroRefCount</code>	This storage unit has a reference count of 0.

GetGenerationNumber - Related Methods

Related Methods

- [ODStorageUnit::IncrementGenerationNumber](#)

GetGenerationNumber - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetID

GetID - Syntax

This method returns the storage-unit ID for this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODID rc;

rc = GetID();
```

GetID Return Value - rc

rc ([ODID](#)) - returns

The storage-unit ID for this storage unit.

GetID - Parameters

rc ([ODID](#)) - returns

The storage-unit ID for this storage unit.

GetID - Exception Handling

GetID - Related Methods

Related Methods

- [ODStorageUnit::GetIDFromStorageUnitRef](#)
-

GetID - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

GetIDFromStorageUnitRef

GetIDFromStorageUnitRef - Syntax

This method returns the storage-unit ID of a referenced storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    aRef;
ODStorageUnitID     rc;

rc = GetIDFromStorageUnitRef(aRef);
```

GetIDFromStorageUnitRef Parameter - aRef

aRef ([ODStorageUnitRef](#)) - input

A persistent storage-unit reference.

GetIDFromStorageUnitRef Return Value - rc

rc ([ODStorageUnitID](#)) - returns
The storage-unit ID of the specified storage unit.

GetIDFromStorageUnitRef - Parameters

aRef ([ODStorageUnitRef](#)) - input
A persistent storage-unit reference.

rc ([ODStorageUnitID](#)) - returns
The storage-unit ID of the specified storage unit.

GetIDFromStorageUnitRef - Remarks

Before you call this method, you must focus this storage unit on the value that created the specified persistent reference. This method looks up the storage unit referenced by the specified persistent reference and returns its storage-unit ID.

GetIDFromStorageUnitRef - Exception Handling

kODErrInvalidStorageUnitRef
kODErrUnfocusedStorageUnit
kODErrZeroRefCount

The specified persistent reference is not valid.
This storage unit is not focused on a value.
This storage unit has a reference count of 0.

GetIDFromStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::GetID](#)
-

GetIDFromStorageUnitRef - Topics

Class:
[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetName

GetName - Syntax

This method returns the name of this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitName rc;

rc = GetName();
```

GetName Return Value - rc

rc ([ODStorageUnitName](#)) - returns
The name of this storage unit or KODNULL if the storage unit does not have a name.

GetName - Parameters

rc ([ODStorageUnitName](#)) - returns
The name of this storage unit or KODNULL if the storage unit does not have a name.

GetName - Remarks

When you no longer need the returned name, you should deallocate it.

GetName - Exception Handling

GetName - Related Methods

Related Methods

- [ODStorageUnit::SetName](#)
-

GetName - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetOffset

GetOffset - Syntax

This method returns the offset of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rc;
```

```
rc = GetOffset();
```

GetOffset Return Value - rc

rc ([ODULong](#)) - returns

The offset, in bytes, of the read/write position from the beginning of the data stream in the currently focused value.

GetOffset - Parameters

rc ([ODULong](#)) - returns

The offset, in bytes, of the read/write position from the beginning of the data stream in the currently focused value.

GetOffset - Remarks

An offset of 0 means the beginning of the data stream corresponding to the focused value; an offset equal to the size returned by the [GetSize](#) method means the end of the data stream.

GetOffset - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetOffset - Related Methods

Related Methods

- [ODStorageUnit::GetSize](#)
 - [ODStorageUnit::SetOffset](#)
-

GetOffset - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetPromiseValue

GetPromiseValue - Syntax

This method reads promise data from the specified value of the currently focused property.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType      valueType;
ODULong          offset;
ODULong          length;
ODByteArray      *value;
ODPart           **sourcePart;
ODULong          rc;

rc = GetPromiseValue(valueType, offset, length,
                    value, sourcePart);
```

GetPromiseValue Parameter - valueType

valueType ([ODValueType](#)) - input
The type of the value from which the promise data is to be read.

GetPromiseValue Parameter - offset

offset ([ODULong](#)) - input
The offset from which the promise data is to be retrieved, from the beginning of the value.

GetPromiseValue Parameter - length

length ([ODULong](#)) - input
The length, in bytes, of the data to be retrieved.

GetPromiseValue Parameter - value

value ([ODByteArray *](#)) - in/out
The byte array whose buffer is to contain the retrieved promise data.

GetPromiseValue Parameter - sourcePart

sourcePart (ODPart **) - output
A reference to the part that made the promise.

GetPromiseValue Return Value - rc

rc (ODULong) - returns
The number of bytes read.

GetPromiseValue - Parameters

valueType (ODValueType) - input
The type of the value from which the promise data is to be read.

offset (ODULong) - input
The offset from which the promise data is to be retrieved, from the beginning of the value.

length (ODULong) - input
The length, in bytes, of the data to be retrieved.

value (ODByteArray *) - in/out
The byte array whose buffer is to contain the retrieved promise data.

sourcePart (ODPart **) - output
A reference to the part that made the promise.

rc (ODULong) - returns
The number of bytes read.

GetPromiseValue - Remarks

You call this method to read promise data without fulfilling the promise. This method first focuses the storage unit on the specified value of the currently focused property. It then starts reading data at the specified offset and stops after reading the number of bytes specified by the *length* parameter or after reaching the end of data in the focused value, whichever comes first.

When you call this method, the *_buffer* field of the *value* output parameter should be KODNULL; if it is not, the buffer to which that field points will not be deallocated.

This method sets the *_buffer* field of the *value* output parameter to point to a memory block containing the promise data, the *_maximum* field to the specified length, and *_length* field to the number of bytes actually read.

This method sets the *sourcePart* parameter to a reference to the part that made the promise. This method does not increment the reference count of the part that made the promise.

When you no longer need the structure you pass the *value* parameter, you should deallocate that structure and its buffer.

GetPromiseValue - Exception Handling

<code>kODErrInvalidType</code>	The specified value type is improperly formed or illegal.
<code>kODErrInvalidValue</code>	This currently focused property does not have a value with the specified value type.
<code>kODErrUnfocusedStorageUnit</code>	This storage unit is not focused on a property or value.
<code>kODErrZeroRefCount</code>	This storage unit has a reference count of 0.

GetPromiseValue - Related Methods

Related Methods

- [ODStorageUnit::IsPromiseValue](#)
 - [ODStorageUnit::SetPromiseValue](#)
-

GetPromiseValue - Topics

Class:

`ODStorageUnit`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetProperty

GetProperty - Syntax

This method returns the name of the property in the current focus context.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODPropertyName rc;
```

```
rc = GetProperty();
```

GetProperty Return Value - rc

rc ([ODPropertyName](#)) - returns
The name of the property in the current focus context.

GetProperty - Parameters

rc ([ODPropertyName](#)) - returns
The name of the property in the current focus context.

GetProperty - Remarks

When you no longer need the returned property name, you should deallocate it.

GetProperty - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetProperty - Related Methods

Related Methods

- [ODStorageUnit::AddProperty](#)
- [ODStorageUnit::Remove](#)

GetProperty - Topics

Class:
ODStorageUnit

Select an item:

GetSession

GetSession - Syntax

This method returns a reference to the session object in which this storage unit runs.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODSession      *rc;

rc = GetSession();
```

GetSession Return Value - rc

rc (ODSession *) - returns
A reference to the session object in which this storage unit runs.

GetSession - Parameters

rc (ODSession *) - returns
A reference to the session object in which this storage unit runs.

GetSession - Exception Handling

kODErrZeroRefCount	This storage unit has a reference count of 0.
--------------------	---

GetSession - Related Methods

Related Methods

- [ODStorageSystem::GetSession](#)

GetSession - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

GetSize

GetSize - Syntax

This method returns the size of the data in the current focus context.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong    rc;

rc = GetSize();
```

GetSize Return Value - rc

rc ([ODULong](#)) - returns

The size, in bytes, of the data in the current focus context.

GetSize - Parameters

rc ([ODULong](#)) - returns

The size, in bytes, of the data in the current focus context.

GetSize - Remarks

If this storage unit is focused on the entire storage unit, this method returns the total size of all properties and values in this storage unit. If it is focused on a property, this method returns the total size of all values in the focused property. If it is focused on a value, this method returns the size of the data stream corresponding to the focused value.

If the currently focused value is a promise value, the promise is fulfilled before the size of the value is evaluated.

GetSize - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetSize - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

GetStrongStorageUnitRef

GetStrongStorageUnitRef - Syntax

This method creates a strong persistence reference to the specified storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitID    embeddedSUID;
ODStorageUnitRef   strongRef;
```

```
GetStrongStorageUnitRef(embeddedSUID, strongRef);
```

GetStrongStorageUnitRef Parameter - embeddedSUID

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

GetStrongStorageUnitRef Parameter - strongRef

strongRef ([ODStorageUnitRef](#)) - output

A persistent reference to the specified storage unit.

GetStrongStorageUnitRef - Return Value

None.

GetStrongStorageUnitRef - Parameters

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

strongRef ([ODStorageUnitRef](#)) - output

A persistent reference to the specified storage unit.

None.

GetStrongStorageUnitRef - Remarks

Before you call this method, you should focus this storage unit on the value where you want to store the strong persistent reference. After this method executes successfully, call the [SetValue](#) method to store the resulting persistent reference, returned in the *strongRef* output parameter, into the currently focused value.

Important: The scope of a persistent reference is limited to the value in which it was created; therefore, when retrieving the storage unit, you must focus it on the same value that the persistent reference was created in.

For more information on persistent references, see the chapter on storage in the *OpenDoc Programming Guide* .

GetStrongStorageUnitRef - Exception Handling

kODErrIllegalNullStorageUnitInput

The *embeddedSUID* parameter is null.

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetStrongStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::GetWeakStorageUnitRef](#)
- [ODStorageUnit::IsStrongStorageUnitRef](#)
- [ODStorageUnit::SetValue](#)

GetStrongStorageUnitRef - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetType

GetType - Syntax

This method returns the type of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODValueType rc;
```

```
rc = GetType();
```

GetType Return Value - rc

rc ([ODValueType](#)) - returns
The type of the currently focused value.

GetType - Parameters

rc ([ODValueType](#)) - returns
The type of the currently focused value.

GetType - Remarks

When you no longer need the returned value type, you should deallocate it.

GetType - Exception Handling

kODErrUnfocusedStorageUnit
kODErrZeroRefCount

This storage unit is not focused on a value.
This storage unit has a reference count of 0.

GetType - Related Methods

Related Methods

- [ODStorageUnit::SetType](#)
-

GetType - Topics

Class:
[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetValue

GetValue - Syntax

This method reads data from the currently focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong      length;
ODByteArray  *value;
ODULong      rc;

rc = GetValue(length, value);
```

GetValue Parameter - length

length (ODULong) - in/out
The length, in bytes, of data to be read.

GetValue Parameter - value

value (ODByteArray *) - in/out
The byte array whose buffer is to contain the retrieved data.

GetValue Return Value - rc

rc (ODULong) - returns
The number of bytes read.

GetValue - Parameters

length (ODULong) - in/out
The length, in bytes, of data to be read.

value (ODByteArray *) - in/out
The byte array whose buffer is to contain the retrieved data.

rc ([ODULong](#)) - returns
The number of bytes read.

GetValue - Remarks

You call this method to read data from the currently focused value. If that value contains a promise value, the promise is fulfilled before the data is read. This method starts reading data at the current offset and stops after reading the number of bytes specified by the *length* parameter or after reaching the end of the data in the currently focused value, whichever comes first.

When you call this method, the *_buffer* field of the *value* output parameter should be `KODNULL`; if it is not, the buffer to which that field points will not be deallocated.

This method sets the *_buffer* field of the *value* output parameter to point to the memory block containing the data that is read from the storage unit; it sets the *_maximum* field to the specified length and the *_length* field to the number of bytes actually read.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

GetValue - Exception Handling

`kODErrUnfocusedStorageUnit`

This storage unit is not focused on a value.

`kODErrZeroRefCount`

This storage unit has a reference count of 0.

GetValue - Related Methods

Related Methods

- [ODStorageUnit::GetSize](#)
 - [ODStorageUnit::SetValue](#)
-

GetValue - Topics

Class:

`ODStorageUnit`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetWeakStorageUnitRef

GetWeakStorageUnitRef - Syntax

This method creates a weak persistent reference to the specified storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    embeddedSUID;
ODStorageUnitRef   weakRef;

GetWeakStorageUnitRef(embeddedSUID, weakRef);
```

GetWeakStorageUnitRef Parameter - embeddedSUID

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

GetWeakStorageUnitRef Parameter - weakRef

weakRef ([ODStorageUnitRef](#)) - output

The persistent reference to the specified storage unit.

GetWeakStorageUnitRef - Return Value

None.

GetWeakStorageUnitRef - Parameters

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

weakRef ([ODStorageUnitRef](#)) - output

The persistent reference to the specified storage unit.

None.

GetWeakStorageUnitRef - Remarks

Before you call this method, you should focus this storage unit on the value where you want to store the weak persistent reference. After this method executes successfully, call the [SetValue](#) method to store the resulting persistent reference, returned in the *weakRef* output parameter, into the currently focused value.

Important: The scope of a persistent reference is limited to the value in which it was created; therefore, when retrieving the storage unit, you must focus it on the same value that the persistent reference was created in.

For more information on persistent references, see the chapter on storage in the *OpenDoc Programming Guide*.

GetWeakStorageUnitRef - Exception Handling

kODErrIllegalNullStorageUnitInput

The *embeddedSUID* parameter is null.

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

GetWeakStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::GetStrongStorageUnitRef](#)
 - [ODStorageUnit::IsWeakStorageUnitRef](#)
 - [ODStorageUnit::SetValue](#)
-

GetWeakStorageUnitRef - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IncrementGenerationNumber

IncrementGenerationNumber - Syntax

This method increments and returns the generation number of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODULong    rc;

rc = IncrementGenerationNumber();
```

IncrementGenerationNumber Return Value - rc

rc ([ODULong](#)) - returns
The generation number of the currently focused value.

IncrementGenerationNumber - Parameters

rc ([ODULong](#)) - returns
The generation number of the currently focused value.

IncrementGenerationNumber - Remarks

You can use the generation number of a value to tell whether the data in the value has changed. For example, when your part makes a significant change to the data in a value, you can call this method to increment its generation number.

IncrementGenerationNumber - Exception Handling

<code>kODErrUnfocusedStorageUnit</code>	This storage unit is not focused on a value.
<code>kODErrZeroRefCount</code>	This storage unit has a reference count of 0.

IncrementGenerationNumber - Related Methods

Related Methods

- [ODStorageUnit::GetGenerationNumber](#)

IncrementGenerationNumber - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

InsertValue

InsertValue - Syntax

This method inserts data into the currently focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *value;

InsertValue(value);
```

InsertValue Parameter - value

value ([ODByteArray *](#)) - input
The byte array whose buffer contains the data to be written.

InsertValue - Return Value

None.

InsertValue - Parameters

value ([ODByteArray *](#)) - input
The byte array whose buffer contains the data to be written.

None.

InsertValue - Remarks

You call this method to insert data into the currently focused value without overwriting the existing data at and beyond the current offset. If the focused value is currently a promise value, the promise is fulfilled before the data is written.

This method writes data to the focused value, starting at the current offset. If the focused value contained any data at and beyond the offset, that data appears after the inserted data. The size of the value is automatically increased to accommodate the inserted data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

InsertValue - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

InsertValue - Related Methods

Related Methods

- [ODStorageUnit::DeleteValue](#)
 - [ODStorageUnit::SetValue](#)
-

InsertValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Internalize

Internalize - Syntax

This method reads all properties and values from this storage unit into memory.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rc;

rc = Internalize();
```

Internalize Return Value - rc

rc (ODStorageUnit *) - returns
A reference to this storage unit.

Internalize - Parameters

rc (ODStorageUnit *) - returns
A reference to this storage unit.

Internalize - Remarks

OpenDoc calls this method; your part does not call this method.

Internalize - Exception Handling

kODErrInvalidStorageUnit

This storage unit is not valid.

Internalize - Related Methods

Related Methods

- [ODStorageUnit::Externalize](#)

Internalize - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IsPromiseValue

IsPromiseValue - Syntax

This method indicates whether the currently focused value is a promise value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rc;
```

```
rc = IsPromiseValue();
```

IsPromiseValue Return Value - rc

rc ([ODBoolean](#)) - returns

A flag indicating whether the currently focused value is a promise value.

kODTrue

The currently focused value is a promise value.

kODFalse

The currently focused value is a regular value.

IsPromiseValue - Parameters

rc ([ODBoolean](#)) - returns

A flag indicating whether the currently focused value is a promise value.

kODTrue

The currently focused value is a promise value.

kODFalse

The currently focused value is a regular value.

IsPromiseValue - Remarks

If the currently focused value is a promise value, the promise is not resolved by this method.

IsPromiseValue - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

IsPromiseValue - Related Methods

Related Methods

- [ODStorageUnit::GetPromiseValue](#)
 - [ODStorageUnit::SetPromiseValue](#)
-

IsPromiseValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IsStrongStorageUnitRef

IsStrongStorageUnitRef - Syntax

This method indicates whether the specified reference is a strong persistent reference.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;
ODBoolean           rc;

rc = IsStrongStorageUnitRef(ref);
```

IsStrongStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input
The persistent reference to be tested. This value is assumed to be valid.

IsStrongStorageUnitRef Return Value - rc

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified reference is a strong persistent reference.

kODTrue	The specified reference is a strong persistent reference.
kODFalse	The specified reference is not a strong persistent reference.

IsStrongStorageUnitRef - Parameters

ref ([ODStorageUnitRef](#)) - input
The persistent reference to be tested. This value is assumed to be valid.

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified reference is a strong persistent reference.

kODTrue	The specified reference is a strong persistent reference.
kODFalse	The specified reference is not a strong persistent reference.

IsStrongStorageUnitRef - Remarks

Before calling this method, you can call the [IsValidStorageUnitRef](#) method to check whether the specified persistent reference is value.

IsStrongStorageUnitRef - Exception Handling

<code>kODErrUnfocusedStorageUnit</code>	This storage unit is not focused on a value.
<code>kODErrZeroRefCount</code>	This storage unit has a reference count of 0.

IsStrongStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::GetStrongStorageUnitRef](#)
- [ODStorageUnit::IsValidStorageUnitRef](#)
- [ODStorageUnit::IsWeakStorageUnitRef](#)

IsStrongStorageUnitRef - Topics

Class:
`ODStorageUnit`

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

IsValidStorageUnitRef

IsValidStorageUnitRef - Syntax

This method indicates whether the specified persistent reference is valid.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
```

```
#include <os2.h>

ODStorageUnitRef    aRef;
ODBoolean           rc;

rc = IsValidStorageUnitRef(aRef);
```

IsValidStorageUnitRef Parameter - aRef

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be tested.

IsValidStorageUnitRef Return Value - rc

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified persistent reference is valid.

kODTrue	The specified persistent reference is valid.
kODFalse	The specified persistent reference is valid.

IsValidStorageUnitRef - Parameters

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be tested.

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified persistent reference is valid.

kODTrue	The specified persistent reference is valid.
kODFalse	The specified persistent reference is valid.

IsValidStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit	This storage unit is not focused on a value.
kODErrZeroRefCount	This storage unit has a reference count of 0.

IsValidStorageUnitRef - Topics

Class:
ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

IsWeakStorageUnitRef

IsWeakStorageUnitRef - Syntax

This method indicates whether the specified persistent reference is a weak persistent reference.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;
ODBoolean           rc;

rc = IsWeakStorageUnitRef(ref);
```

IsWeakStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input

The persistent reference to be tested. This avalue is assumed to be valid.

IsWeakStorageUnitRef Return Value - rc

rc ([ODBoolean](#)) - returns

A flag indicating whether the specified reference is a weak persistent reference.

kODTrue

The specified reference is a weak persistent reference.

kODFalse

The specified reference is not a weak persistent reference.

IsWeakStorageUnitRef - Parameters

ref ([ODStorageUnitRef](#)) - input
The persistent reference to be tested. This avalue is assumed to be valid.

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified reference is a weak persistent reference.

kODTrue	The specified reference is a weak persistent reference.
kODFalse	The specified reference is not a weak persistent reference.

IsWeakStorageUnitRef - Remarks

Before calling this method, you can call the [IsValidStorageUnitRef](#) method to check whether the specified persistent reference is valid.

IsWeakStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit	This storage unit is not focused on a value.
kODErrZeroRefCount	This storage unit has a reference count of 0.

IsWeakStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::GetWeakStorageUnitRef](#)
 - [ODStorageUnit::IsStrongStorageUnitRef](#)
 - [ODStorageUnit::IsValidStorageUnitRef](#)
-

IsWeakStorageUnitRef - Topics

Class:
[ODStorageUnit](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

Lock

Lock - Syntax

This method locks this storage unit for exclusive access.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitKey    key;
ODStorageUnitKey    rc;

rc = Lock(key);
```

Lock Parameter - key

key ([ODStorageUnitKey](#)) - input

The previously acquired identifier for a particular clone operation, or the value `KODNULLKey` if the valid storage unit key is unknown.

Lock Return Value - rc

rc ([ODStorageUnitKey](#)) - returns

The identifier for the locked state of this storage unit.

Lock - Parameters

key ([ODStorageUnitKey](#)) - input

The previously acquired identifier for a particular clone operation, or the value `KODNULLKey` if the valid storage unit key is unknown.

rc ([ODStorageUnitKey](#)) - returns

The identifier for the locked state of this storage unit.

Lock - Remarks

Every thread must acquire the storage unit key for multithreading support.

Lock - Exception Handling

KODerrInvalidStorageUnitKey

The specified storage unit key is not valid.

Lock - Related Methods

Related Methods

- [ODStorageUnit::Unlock](#)

Lock - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Remove

Remove - Syntax

This method removes all properties and values in the current current focus context from this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnit      *rc;
```

```
rc = Remove();
```

Remove Return Value - rc

rc (ODStorageUnit *) - returns

A reference to this storage unit.

Remove - Parameters

rc (ODStorageUnit *) - returns
A reference to this storage unit.

Remove - Remarks

If the current focus context is the entire storage unit, this method removes all properties and their values. If it is focused on a property, this method removes the focused property and all its values. If it is focused on a value, this method removes the focused value.

After this method executes successfully, the storage unit is unfocused.

Remove - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

Remove - Related Methods

Related Methods

- [ODStorageUnit::AddProperty](#)
 - [ODStorageUnit::AddValue](#)
-

Remove - Topics

Class:
ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

RemoveStorageUnitRef

RemoveStorageUnitRef - Syntax

This method makes a persistent reference invalid in the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    aRef;
ODStorageUnit       *rc;

rc = RemoveStorageUnitRef(aRef);
```

RemoveStorageUnitRef Parameter - aRef

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be removed.

RemoveStorageUnitRef Return Value - rc

rc (ODStorageUnit *) - returns
A reference to this storage unit.

RemoveStorageUnitRef - Parameters

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be removed.

rc (ODStorageUnit *) - returns
A reference to this storage unit.

RemoveStorageUnitRef - Remarks

This method does not change the data in the currently focused value, but after this method is called, the specified persistent reference is no longer valid. To remove data corresponding to the persistent reference, you must call the [DeleteValue](#) method.

RemoveStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

RemoveStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnit::DeleteValue](#)

RemoveStorageUnitRef - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

ResolveAllPromises

ResolveAllPromises - Syntax

This method resolves all promises in this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ResolveAllPromises();
```

ResolveAllPromises - Return Value

None.

ResolveAllPromises - Parameters

None.

ResolveAllPromises - Remarks

To resolve a promise, the storage unit calls the [FulfillPromise](#) method of the part that made the promise. This method does not change the current focus context.

ResolveAllPromises - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

ResolveAllPromises - Related Methods

Related Methods

- [ODPart::FulfillPromise](#)
 - [ODStorageUnit::ClearAllPromises](#)
-

ResolveAllPromises - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetName

SetName - Syntax

This method sets the name of this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitName    name;

SetName (name);
```

SetName Parameter - name

name ([ODStorageUnitName](#)) - input
The name to be assigned to this storage unit.

SetName - Return Value

None.

SetName - Parameters

name ([ODStorageUnitName](#)) - input
The name to be assigned to this storage unit.

None.

SetName - Exception Handling

kODErrZeroRefCount

This storage unit has a reference count of 0.

SetName - Related Methods

Related Methods

- [ODStorageUnit::GetName](#)
-

SetName - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

SetOffset

SetOffset - Syntax

This method sets the offset of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    offset;
```

```
SetOffset(offset);
```

SetOffset Parameter - offset

offset ([ODULong](#)) - input

The new offset, in bytes, of the read/write position from the beginning of the data stream in the currently focused value.

SetOffset - Return Value

None.

SetOffset - Parameters

offset ([ODULong](#)) - input

The new offset, in bytes, of the read/write position from the beginning of the data stream in the currently focused value.

None.

SetOffset - Remarks

You can call this method if you want to read or write data at a particular position in the focused value. An offset of 0 means the beginning of the data stream corresponding to the focused value; an offset equal to the current size of the focused value (as returned by the [GetSize](#) method) means the end of the data stream. You may not specify an offset larger than the current size of the focused value.

SetOffset - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

SetOffset - Related Methods

Related Methods

- [ODStorageUnit::GetOffset](#)
 - [ODStorageUnit::GetSize](#)
-

SetOffset - Topics

Class:

ODStorageUnit

Select an item:

[Syntax](#)

[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetPromiseValue

SetPromiseValue - Syntax

This method writes data to the specified value of the currently focused property, creating the value if it does not exist and making the value a promise value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType      valueType;
ODULong          offset;
ODByteArray      *value;
ODPart           *sourcePart;

SetPromiseValue(valueType, offset, value,
                sourcePart);
```

SetPromiseValue Parameter - valueType

valueType ([ODValueType](#)) - input
The type of the value where the promise data is to be written.

SetPromiseValue Parameter - offset

offset ([ODULong](#)) - input
The offset at which the promise data is to be stored, from the beginning of the value.

SetPromiseValue Parameter - value

value ([ODByteArray *](#)) - input
The byte array whose buffer contains the promise data to be written.

SetPromiseValue Parameter - sourcePart

sourcePart (ODPart *) - input
A reference to the part that made the promise.

SetPromiseValue - Return Value

None.

SetPromiseValue - Parameters

valueType (ODValueType) - input
The type of the value where the promise data is to be written.

offset (ODULong) - input
The offset at which the promise data is to be stored, from the beginning of the value.

value (ODByteArray *) - input
The byte array whose buffer contains the promise data to be written.

sourcePart (ODPart *) - input
A reference to the part that made the promise.

None.

SetPromiseValue - Remarks

You call this method to write a promise for a value of the specified type in the currently focused property. You may call this method multiple times to promise values of different types or to write to different offsets in the same value.

This method writes data to the specified value, starting at the specified offset (inclusive), and overwrites any data at and beyond the offset. If the current offset plus the length of the data being written is greater than the current size of the value (as returned by the [GetSize](#) method), the size of the value is automatically increased to accomodate the new data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

SetPromiseValue - Exception Handling

kODErrInvalidType
kODErrCannotAddType
kODErrUnfocusedStorageUnit

The specified value type is null.
The specified value type is improperly formed or illegal.
This storage unit is not focused on a property or value.

SetPromiseValue - Related Methods

Related Methods

- [ODStorageUnit::GetPromiseValue](#)
 - [ODStorageUnit::GetSize](#)
 - [ODStorageUnit::IsPromiseValue](#)
-

SetPromiseValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

SetStorageUnitRef

SetStorageUnitRef - Syntax

This method creates, in the currently focused value, a specified persistent reference to to specified storage unit using the specified persistent identifier.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    embeddedSUID;
ODStorageUnitRef   ref;

SetStorageUnitRef(embeddedSUID, ref);
```

SetStorageUnitRef Parameter - embeddedSUID

embeddedSUID ([ODStorageUnitID](#)) - input
The storage-unit ID for the storage unit to be referenced.

SetStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input
A persistent identifier for the new persistent reference being created.

SetStorageUnitRef - Return Value

None.

SetStorageUnitRef - Parameters

embeddedSUID ([ODStorageUnitID](#)) - input
The storage-unit ID for the storage unit to be referenced.

ref ([ODStorageUnitRef](#)) - input
A persistent identifier for the new persistent reference being created.

None.

SetStorageUnitRef - Remarks

This method is called only by the container suite. Parts, the document shell, and container applications should call this method.

The *embeddedSUID* parameter specifies the ID that identifies the storage unit within the current session. The *ref* parameter specifies the reference to be used within the currently focused value to identify the storage unit persistently across sessions.

SetStorageUnitRef - Exception Handling

kODErrInvalidID
kODErrUnfocusedStorageUnit

The *embeddedSUID* parameter is invalid.
This storage unit is not focused on a property and value.

SetStorageUnitRef - Topics

Class:
ODStorageUnit

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

SetType

SetType - Syntax

This method sets the type of the currently focused value.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType    valueType;

SetType(valueType);
```

SetType Parameter - valueType

valueType ([ODValueType](#)) - input
The new type for the currently focused value.

SetType - Return Value

None.

SetType - Parameters

valueType ([ODValueType](#)) - input
The new type for the currently focused value.

None.

SetType - Exception Handling

kODErrInvalidType
kODErrUnfocusedStorageUnit
kODErrZeroRefCount

The specified value type is improperly formed or illegal.
This storage unit is not focused on a value.
This storage unit has a reference count of 0.

SetType - Related Methods

Related Methods

- [ODStorageUnit::GetType](#)
-

SetType - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Exception Handling](#)
[Related Methods](#)

SetValue

SetValue - Syntax

This method writes data to the currently focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *value;

SetValue(value);
```

SetValue Parameter - value

value ([ODByteArray *](#)) - input
A bytes array whose buffer contains the data to be written.

SetValue - Return Value

None.

SetValue - Parameters

value ([ODByteArray *](#)) - input
A bytes array whose buffer contains the data to be written.

None.

SetValue - Remarks

You call this method to write data to the currently focused value. If that value currently is a promise value, the promise is fulfilled before the data is written.

This method writes data to the focused value, starting at the current offset, and overwrites any data at and beyond the offset. If the current offset plus the length of data being written is greater than the current size of the value (as returned by the [GetSize](#) method), the size of the value is automatically increased to accommodate the new data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

SetValue - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a value.

kODErrZeroRefCount

This storage unit has a reference count of 0.

SetValue - Related Methods

Related Methods

- [ODStorageUnit::GetSize](#)
 - [ODStorageUnit::GetValue](#)
-

SetValue - Topics

Class:

[ODStorageUnit](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Unlock

Unlock - Syntax

This method unlocks this storage unit.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitKey    key;
```

```
Unlock(key);
```

Unlock Parameter - key

key ([ODStorageUnitKey](#)) - input

The storage unit key acquired from the [Lock](#) method.

Unlock - Return Value

None.

Unlock - Parameters

key ([ODStorageUnitKey](#)) - input
The storage unit key acquired from the [Lock](#) method.

None.

Unlock - Remarks

Every thread must acquire the storage unit key for multithreading support.

Unlock - Exception Handling

kODErrInvalidStorageUnitKey
kODErrStorageUnitNotLocked

The specified storage unit key is not valid.
This storage unit is not locked.

Unlock - Related Methods

Related Methods

- [ODStorageUnit::Lock](#)
-

Unlock - Topics

Class:
[ODStorageUnit](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

ODStorageUnitCursor

Class Definition File: SUCURSOR.IDL

Class Hierarchy

SOMObject
ODObject
 ODStorageUnitCursor

Description

An object of the ODStorageUnitCursor class provides swift focusing on frequently accessed data in a storage unit.

A storage-unit cursor represents a *focus context* for a storage unit. The focus context can be the entire storage unit, a particular property, or a particular value of a particular property. When the focus context is the entire storage unit, the data of interest includes all properties and all their values. When the focus context is a particular property, the data of interest includes all values of that property. When the focus context is a particular value, the data of interest is the data stream corresponding to that value.

A storage-unit cursor uses a property name, a value type, and a value index to specify its focus context. Methods of the storage-unit cursor allow you to get and set each of these three pieces of information.

- To specify the entire storage unit as the focus context, set the property name to kODNULL; set the value type to kODTypeAll and value index to kODIndexAll.
- To specify a property as the focus context, set the property name to the name of the desired property; set the value type to kODTypeAll and value index to kODIndexAll.
- To specify a value as the focus context, set the property name to the name of the property containing the value. You can specify the value by either its type or its index:
 - To use its type, set the value type to the type of the desired value and the value index to 0.
 - To use its index, set the value type to kODNULL and the value index to the index of the desired value.

A storage-unit cursor makes it simple for you to focus the storage unit on the corresponding focus context. Your part creates a storage-unit cursor object by calling its storage unit's [CreateCursor](#) or [CreateCursorWithFocus](#) methods. Your part calls its storage unit's [FocusWithCursor](#) method to focus the storage unit on the focus context represented by a storage-unit cursor.

For more information about storage units, focus context, and value indexes, see the class description for [ODStorageUnit](#).

Methods

The methods defined by the ODStorageUnitCursor class include:

- [GetProperty](#)
- [GetValueIndex](#)
- [GetValueType](#)
- [SetProperty](#)
- [SetValueIndex](#)
- [SetValueType](#)

Overridden Methods

There are currently no methods overridden by the ODStorageUnitCursor class.

GetProperty

GetProperty - Syntax

This method gets the property name of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
```

```
#include <os2.h>

ODPropertyName      *propertyName;

GetProperty(propertyName);
```

GetProperty Parameter - propertyName

propertyName ([ODPropertyName *](#)) - output
The name of the property in the focus context.

GetProperty - Return Value

None.

GetProperty - Parameters

propertyName ([ODPropertyName *](#)) - output
The name of the property in the focus context.

None.

GetProperty - Remarks

When you no longer need the property name returned in the *propertyName* parameter, you should deallocate it.

GetProperty - Related Methods

Related Methods

- [ODStorageUnitCursor::SetProperty](#)
-

GetProperty - Topics

Class:
ODStorageUnitCursor

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

GetValueIndex

GetValueIndex - Syntax

This method gets the value index of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueIndex      *valueIndex;

GetValueIndex(valueIndex);
```

GetValueIndex Parameter - valueIndex

valueIndex ([ODValueIndex *](#)) - output
The value index in the focus context.

GetValueIndex - Return Value

None.

GetValueIndex - Parameters

valueIndex ([ODValueIndex *](#)) - output
The value index in the focus context.

None.

GetValueIndex - Related Methods

Related Methods

- [ODStorageUnitCursor::SetValueIndex](#)
-

GetValueIndex - Topics

Class:

ODStorageUnitCursor

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

GetValueType

GetValueType - Syntax

This method gets the value type of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType      *valueType;

GetValueType(valueType);
```

GetValueType Parameter - valueType

valueType ([ODValueType](#) *) - output
The value type of the focus context.

GetValueType - Return Value

None.

GetValueType - Parameters

valueType ([ODValueType *](#)) - output
The value type of the focus context.

None.

GetValueType - Remarks

When you no longer need the value type returned in the *valueType* parameter, you should deallocate it.

GetValueType - Related Methods

Related Methods

- [ODStorageUnitCursor::SetValueType](#)
-

GetValueType - Topics

Class:

[ODStorageUnitCursor](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

SetProperty

SetProperty - Syntax

This method sets the property name of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODPropertyName    propertyName;

SetProperty(propertyName);
```

SetProperty Parameter - propertyName

propertyName ([ODPropertyName](#)) - input

The name of the property in the focus context, or KODNULL if the focus context is the entire storage unit.

SetProperty - Return Value

None.

SetProperty - Parameters

propertyName ([ODPropertyName](#)) - input

The name of the property in the focus context, or KODNULL if the focus context is the entire storage unit.

None.

SetProperty - Remarks

When you no longer need the property name you pass as the *propertyName* parameter, you should deallocate it.

SetProperty - Related Methods

Related Methods

- [ODStorageUnitCursor::GetProperty](#)

SetProperty - Topics

Class:

ODStorageUnitCursor

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

SetValueIndex

SetValueIndex - Syntax

This method sets the value index of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueIndex    valueIndex;

SetValueIndex(valueIndex);
```

SetValueIndex Parameter - valueIndex

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the focus context, or 0 to ignore the value index (and use the value type to specify the value in the focus context).

SetValueIndex - Return Value

None.

SetValueIndex - Parameters

valueIndex ([ODValueIndex](#)) - input

The value index of the value in the focus context, or 0 to ignore the value index (and use the value type to specify the value in the focus context).

None.

SetValueIndex - Remarks

When you use this storage-unit cursor to focus your part's storage unit, the value index of this storage-unit cursor is ignored unless the value type is KODNULL.

SetValueIndex - Related Methods

Related Methods

- [ODStorageUnitCursor::GetValueIndex](#)
-

SetValueIndex - Topics

Class:

[ODStorageUnitCursor](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

SetValueType

SetValueType - Syntax

This method sets the value type of this storage-unit cursor.

```
#define INCL_ODSTORAGEUNIT
#define INCL_ODAPI
#include <os2.h>

ODValueType    valueType;

SetValueType(valueType);
```

SetValueType Parameter - valueType

valueType ([ODValueType](#)) - input

The value type of the value in the focus context, or KODNULL to ignore the value type (and use the value index to specify the value in the focus context).

SetValueType - Return Value

None.

SetValueType - Parameters

valueType ([ODValueType](#)) - input

The value type of the value in the focus context, or KODNULL to ignore the value type (and use the value index to specify the value in the focus context).

None.

SetValueType - Remarks

When you no longer need the value type you pass as the *valueType* parameter, you should deallocate it.

SetValueType - Related Methods

Related Methods

- [ODStorageUnitCursor::GetValueType](#)

SetValueType - Topics

Class:

ODStorageUnitCursor

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

ODStorageUnitRefIterator

Class Definition File: SUREFTR.IDL

Class Hierarchy

SOMObject

ODObject

ODStorageUnitRefIterator

Description

An object of the ODStorageUnitRefIterator class provides access to all persistent references in a currently focused value.

If a value in a storage unit contains persistent references, storage-unit reference iterator can provide access to each of those persistent references. For example, a caller might need to access all referenced storage units in a clone operation. A caller can also use a storage-unit reference iterator to access the persistent reference in any storage unit value. This practice might be helpful for utilities that fix up cross-references or index a document. Persistent references cannot be created or removed during iteration, and the storage unit must remain focused to the value on which the storage-unit reference iterator is iterating.

Callers create a storage-unit reference iterator object by calling its storage unit's [CreateStorageUnitRefIterator](#) method, which returns a reference to a storage-unit reference iterator object.

For more information on cloning, see the chapters on storage and data transfer in the *OpenDoc Programming Guide* . For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

Methods

The methods defined by the ODStorageUnitRefIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

Overridden Methods

There are currently no methods overridden by the ODStorageUnitRefIterator class.

First

First - Syntax

This method begins the iteration and returns a copy of the first persistent reference, if it exists, in the currently focused value of the storage

unit.

```
#define INCL_ODSTORAGEUNITREFITERATOR
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;

First(ref);
```

First Parameter - ref

ref ([ODStorageUnitRef](#)) - output

A copy of the first persistent reference in the currently focused value of the storage unit. If the focused value contains no persistent references, the return value is undefined.

First - Return Value

None.

First - Parameters

ref ([ODStorageUnitRef](#)) - output

A copy of the first persistent reference in the currently focused value of the storage unit. If the focused value contains no persistent references, the return value is undefined.

None.

First - Remarks

Your part must call this method before calling the [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

It is your responsibility to deallocate the returned persistent reference when it is no longer needed.

First - Exception Handling

kODErrIteratorOutOfSync

The focused value was modified while the iteration was in progress.

First - Topics

Class:
ODStorageUnitReflterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

IsNotComplete

IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODSTORAGEUNITREFITERATOR
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsNotComplete();
```

IsNotComplete Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the iteration is incomplete

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

IsNotComplete - Parameters

rv ([ODBoolean](#)) - returns
A flag indicating whether the iteration is incomplete

kODTrue

kODFalse	The iteration is incomplete.
	The iteration is complete.

IsNotComplete - Remarks

Your part calls this method to test whether more persistent references remain in the focused valued. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a persietent reference. This method returns kODFalse when you have examinedd all the persistent references (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

IsNotComplete - Exception Handling

kODErrIteratorNotInitialized	This method was called before calling either the First or Next method to begin the iteration.
kODErrIteratorOutOfSync	The focused value was modified while the iteration was in progress.

IsNotComplete - Topics

Class:
ODStorageUnitRefIterator

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

Next

Next - Syntax

This method returns a copy of the next persistent reference, if it exists, in the currently focused value of the storage unit.

```
#define INCL_ODSTORAGEUNITREFITERATOR
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;

Next (ref);
```

Next Parameter - ref

ref ([ODStorageUnitRef](#)) - output

A copy of the next persistent reference in the currently focused value of the storage unit. If the focused value contains no persistent references or if the iteration is complete, the return value is undefined.

Next - Return Value

None.

Next - Parameters

ref ([ODStorageUnitRef](#)) - output

A copy of the next persistent reference in the currently focused value of the storage unit. If the focused value contains no persistent references or if the iteration is complete, the return value is undefined.

None.

Next - Remarks

If your part calls this method before calling this storage-unit reference iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

It is your responsibility to deallocate the returned persistent reference when it is no longer needed.

Next - Exception Handling

`kODErrIteratorOutOfSync`

The focused value was modified while the iteration was in progress.

Next - Topics

Class:

`ODStorageUnitRefIterator`

Select an item:

ODStorageUnitView

Class Definition File: SUVIEW.IDL

Class Hierarchy

SOMObject
 ODObject
 ODStorageUnitView

Description

An object of the ODStorageUnitView class provides thread-safe access to a storage unit by automatically focusing and locking the storage unit.

A storage-unit view represents a particular storage unit, prefocused on a particular focus context. Your part creates a storage-unit view object by focusing a storage unit and then calling the [CreateView](#) method of that storage unit. You can pass the storage-unit view among your software components to give them access to the particular focused data stream.

A storage-unit view has most of the functionality of a storage unit, except that it has no methods for changing the focus. When you access a storage unit through a storage-unit view, however, the storage-unit view automatically locks the storage unit during the access and unlocks it afterward.

The storage-unit view has an associated storage-unit cursor that represents the focus context of the storage-unit view. You can call the [GetCursor](#) method of a storage-unit view to obtain its storage-unit cursor. If you make changes to that storage-unit cursor, you change the focus context of the storage-unit view. Typically, however, parts do not change the focus context of a storage-unit view.

For more information related to storage units and storage-unit cursors, see the class descriptions for [ODStorageUnit](#) and [ODStorageUnitCursor](#).

Methods

The methods defined by the ODStorageUnitView class include:

Storage

- [CloneInto](#)
- [Externalize](#)
- [Internalize](#)

Manipulating Storage Units

- [AddProperty](#)
- [AddValue](#)
- [GetID](#)
- [GetName](#)
- [GetSize](#)
- [GetStorageUnit](#)
- [Remove](#)
- [SetName](#)

Accessing Focus Context

- [GetCursor](#)
- [GetProperty](#)

Manipulating Value

- [DeleteValue](#)
- [GetGenerationNumber](#)
- [GetOffset](#)
- [GetType](#)
- [GetValue](#)
- [IncrementGenerationNumber](#)
- [InsertValue](#)

- [SetOffset](#)
- [SetType](#)
- [SetValue](#)

Manipulating Persistent Reference

- [GetIDFromStorageUnitRef](#)
- [GetStrongStorageUnitRef](#)
- [GetWeakStorageUnitRef](#)
- [IsStrongStorageUnitRef](#)
- [IsValidStorageUnitRef](#)
- [IsWeakStorageUnitRef](#)
- [RemoveStorageUnitRef](#)

Manipulating Promise

- [GetPromiseValue](#)
- [IsPromiseValue](#)
- [SetPromiseValue](#)

Creating Objects

- [CreateStorageUnitRefIterator](#)

Overridden Methods

There are currently no methods overridden by the ODStorageUnitView class.

AddProperty

AddProperty - Syntax

This method adds a property with the specified name to the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODPropertyName      propertyName;
ODStorageUnitView   *rv;

rv = AddProperty(propertyName);
```

AddProperty Parameter - propertyName

propertyName ([ODPropertyName](#)) - input
The name of the property to be added.

AddProperty Return Value - rv

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

AddProperty - Parameters

propertyName ([ODPropertyName](#)) - input
The name of the property to be added.

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

AddProperty - Remarks

If the storage unit that created this storage-unit view does not already contain a property with the specified name, the new property is added and the storage unit is focused on the newly added property. Otherwise, the storage unit is focused on the existing property with the specified name.

The focus context of this storage-unit view remains unchanged.

AddProperty - Exception Handling

kODErrCannotAddProperty

The specified property cannot be added to the storage unit that created this storage-unit view.

kODErrIllegalNullPropertyInput

The specified property name is null.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

AddProperty - Related Methods

Related Methods

- [ODStorageUnitView::GetProperty](#)
 - [ODStorageUnitView::Remove](#)
-

AddProperty - Topics

Class:
ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

AddValue

AddValue - Syntax

This method adds a value of the specified type to the focused property.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODValueType          type;
ODStorageUnitView    *rv;

rv = AddValue(type);
```

AddValue Parameter - type

type ([ODValueType](#)) - input
The type of value to be added.

AddValue Return Value - rv

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

AddValue - Parameters

type ([ODValueType](#)) - input
The type of value to be added.

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

AddValue - Remarks

If the focused property does not already contain a value of the specified type, the new value is added and the storage unit that created that storage-unit view is focused on the newly added value. Otherwise, the storage unit is focused on the existing value with the specified value type.

The focus context of this storage-unit view remains unchanged.

AddValue - Exception Handling

kODErrInvalidType

The specified value type is improperly formed or illegal.

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a property or a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

AddValue - Related Methods

Related Methods

- [ODStorageUnitView::Remove](#)
-

AddValue - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CloneInto

CloneInto - Syntax

This method copies all properties and values of the storage unit that created this storage-unit view into the specified destination storage unit.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODDraftKey      key;
ODStorageUnit   *destStorageUnit;
ODID            scopeID;

CloneInto(key, destStorageUnit, scopeID);
```

CloneInto Parameter - key

key (ODDraftKey) - input
The draft key identifying this cloning operation.

CloneInto Parameter - destStorageUnit

destStorageUnit (ODStorageUnit *) - input
A reference to the destination storage unit to which the data is to be copied.

CloneInto Parameter - scopeID

scopeID (ODID) - input
The ID of the frame that defines the scope of this cloning operation or kODIDAll if all referenced objects are within scope.

CloneInto - Return Value

None.

CloneInto - Parameters

key (ODDraftKey) - input
The draft key identifying this cloning operation.

destStorageUnit (ODStorageUnit *) - input

A reference to the destination storage unit to which the data is to be copied.

scopeID ([ODID](#)) - input

The ID of the frame that defines the scope of this cloning operation or kODIDAll if all referenced objects are within scope.

None.

CloneInto - Remarks

This method is not called by parts. Your part typically calls its draft's [Clone](#) method or [WeakClone](#) method instead of this method.

If this storage unit that created this storage-unit view has persistent references to other objects, the *scopeID* parameter determines which of the referenced objects are within the scope of this cloning operation. Typically, the *scopeID* parameter is the ID of a frame, and only those objects embedded in that frame are within scope.

The storage unit that created this storage-unit view is the source storage unit for the cloning operation. This method copies the source storage unit's data into the specified destination storage unit. If the source storage unit has persistent references to other objects, this method clones any additional persistent referenced objects that are within the scope of this cloning operation. Objects referenced by strong persistent references are strongly cloned by recursive calls to the [Clone](#) method; objects referenced by weak persistent references are weakly cloned by calls to the [WeakClone](#) method.

CloneInto - Exception Handling

kODErrInvalidDraftKey

The specified draft key is not the draft key for the current cloning operation.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

CloneInto - Related Methods

Related Methods

- [ODDraft::Clone](#)
- [ODDraft::WeakClone](#)

CloneInto - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

CreateStorageUnitRefIterator

CreateStorageUnitRefIterator - Syntax

This method creates a storage-unit reference iterator for the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRefIterator    *rv;

rv = CreateStorageUnitRefIterator();
```

CreateStorageUnitRefIterator Return Value - rv

rv (ODStorageUnitRefIterator *) - returns

A reference to the storage-unit reference iterator used to traverse all persistent references in the focused value.

CreateStorageUnitRefIterator - Parameters

rv (ODStorageUnitRefIterator *) - returns

A reference to the storage-unit reference iterator used to traverse all persistent references in the focused value.

CreateStorageUnitRefIterator - Remarks

You can call this method if you need to perform an operation on all storage-unit references in the focused value.

While you are using the returned storage-unit reference iterator, you must not modify the focused value; in particular, you must not call any of the following methods of this storage unit: [DeleteValue](#), [Internalize](#), [Remove](#), [RemoveStorageUnitRef](#), [SetType](#), or [SetValue](#). Furthermore, you must not delete this storage-unit view.

You must delete the returned storage-unit reference iterator when it is no longer needed.

CreateStorageUnitRefIterator - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

CreateStorageUnitReflterator - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

DeleteValue

DeleteValue - Syntax

This method deletes data from the currently focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    length;
```

```
DeleteValue(length);
```

DeleteValue Parameter - length

length ([ODULong](#)) - input

The number of bytes of data to be deleted.

DeleteValue - Return Value

None.

DeleteValue - Parameters

length ([ODULong](#)) - input
The number of bytes of data to be deleted.

None.

DeleteValue - Remarks

You call this method to delete data from the currently focused value. If that value is a promise value, the promise is fulfilled before the data is deleted. This method starts deleting data at the current offset and stops after deleting the number of bytes specified by the *length* parameter or after reaching the end of the data in the currently focused value, whichever comes first.

DeleteValue - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

DeleteValue - Related Methods

Related Methods

- [ODStorageUnitView::InsertValue](#)
 - [ODStorageUnitView::Remove](#)
-

DeleteValue - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Externalize

Externalize - Syntax

This method resolves all promises in the storage unit that created this storage-unit view and saves all its properties and values to persistent, external storage.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView      *rv;

rv = Externalize();
```

Externalize Return Value - rv

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Externalize - Parameters

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Externalize - Exception Handling

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

Externalize - Related Methods

Related Methods

- [ODStorageUnitView::Internalize](#)

Externalize - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Exception Handling](#)[Related Methods](#)

GetCursor

GetCursor - Syntax

This method returns a reference to the storage-unit cursor representing the focus context of this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitCursor    *rv;
```

```
rv = GetCursor();
```

GetCursor Return Value - rv

rv (ODStorageUnitCursor *) - returns

A reference to the storage-unit cursor representing the focus context of this storage-unit view.

GetCursor - Parameters

rv (ODStorageUnitCursor *) - returns

A reference to the storage-unit cursor representing the focus context of this storage-unit view.

GetCursor - Remarks

You can change the focus context of this storage-unit view by modifying the returned storage-unit cursor; typically, however, parts do not change the focus context of the storage-unit view.

You should not delete the returned storage-unit cursor.

GetCursor - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

GetGenerationNumber

GetGenerationNumber - Syntax

This method returns the generation number of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetGenerationNumber();
```

GetGenerationNumber Return Value - rv

rv ([ODULong](#)) - returns

The generation number of the focused value.

GetGenerationNumber - Parameters

rv ([ODULong](#)) - returns

The generation number of the focused value.

GetGenerationNumber - Remarks

You can use the generation number of a value to tell whether the data in the value has changed. For example, your part could compare the number returned by this method with a saved generation number.

GetGenerationNumber - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetGenerationNumber - Related Methods

Related Methods

- [ODStorageUnitView::IncrementGenerationNumber](#)
-

GetGenerationNumber - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetID

GetID - Syntax

This method returns the storage-unit ID for the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODID    rv;
```

```
rv = GetID();
```

GetID Return Value - rv

rv ([ODID](#)) - returns
The storage-unit ID for the storage unit that created this storage-unit view.

GetID - Parameters

rv ([ODID](#)) - returns
The storage-unit ID for the storage unit that created this storage-unit view.

GetID - Related Methods

Related Methods

- [ODStorageUnitView::GetIDFromStorageUnitRef](#)
-

GetID - Topics

Class:
ODStorageUnitView

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Related Methods](#)

GetIDFromStorageUnitRef

GetIDFromStorageUnitRef - Syntax

This method returns the storage-unit ID of a referenced storage unit.

```

#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    aRef;
ODStorageUnitID     rv;

rv = GetIDFromStorageUnitRef(aRef);

```

GetIDFromStorageUnitRef Parameter - aRef

aRef ([ODStorageUnitRef](#)) - input
A persistent storage-unit reference.

GetIDFromStorageUnitRef Return Value - rv

rv ([ODStorageUnitID](#)) - returns
The storage-unit ID of the storage unit referenced by the specified persistent reference.

GetIDFromStorageUnitRef - Parameters

aRef ([ODStorageUnitRef](#)) - input
A persistent storage-unit reference.

rv ([ODStorageUnitID](#)) - returns
The storage-unit ID of the storage unit referenced by the specified persistent reference.

GetIDFromStorageUnitRef - Remarks

The focused value must be the same value that created the specified persistent reference. This method looks up the storage unit referenced by the specified persistent reference and returns its storage-unit ID.

GetIDFromStorageUnitRef - Exception Handling

kODErrInvalidStorageUnitRef

The specified persistent reference is not valid.

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetIDFromStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::GetID](#)

GetIDFromStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetName

GetName - Syntax

This method returns the name of the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitName    rv;
```

```
rv = GetName();
```

GetName Return Value - rv

rv ([ODStorageUnitName](#)) - returns

The name of the storage unit that created this storage-unit view or KODNULL if the storage unit does not have a name.

GetName - Parameters

rv ([ODStorageUnitName](#)) - returns

The name of the storage unit that created this storage-unit view or KODNULL if the storage unit does not have a name.

GetName - Remarks

When you no longer need the returned name, you should deallocate it.

GetName - Related Methods

Related Methods

- [ODStorageUnitView::SetName](#)
-

GetName - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

GetOffset

GetOffset - Syntax

This method returns the offset of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetOffset();
```

GetOffset Return Value - rv

rv ([ODULong](#)) - returns

The offset, in bytes, of the read/write position from the beginning of the data stream in the focused value.

GetOffset - Parameters

rv ([ODULong](#)) - returns

The offset, in bytes, of the read/write position from the beginning of the data stream in the focused value.

GetOffset - Remarks

An offset of 0 means the beginning of the data stream corresponding to the focused value; an offset equal to the size returned by the [GetSize](#) method means the end of the data stream.

GetOffset - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetOffset - Related Methods

Related Methods

- [ODStorageUnitView::GetSize](#)
 - [ODStorageUnitView::SetOffset](#)
-

GetOffset - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetPromiseValue

GetPromiseValue - Syntax

This method reads the promise data from the specified value of the focused property.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODValueType      valueType;
ODULong          offset;
ODULong          length;
ODByteArray      *value;
ODPart           **sourcePart;
ODULong          rv;

rv = GetPromiseValue(valueType, offset, length,
                    value, sourcePart);
```

GetPromiseValue Parameter - valueType

valueType ([ODValueType](#)) - input
The type of the value from which the promise data is to be read.

GetPromiseValue Parameter - offset

offset ([ODULong](#)) - input
The offset from which the promise data is to be retrieve, from the beginning of the value.

GetPromiseValue Parameter - length

length ([ODULong](#)) - input
The length, in bytes, of data to be retrieved.

GetPromiseValue Parameter - value

value (ODByteArray *) - input
The byte array whose buffer contains the retrieved promise data.

GetPromiseValue Parameter - sourcePart

sourcePart (ODPart **) - output
A reference to the part that made the promise.

GetPromiseValue Return Value - rv

rv (ODULong) - returns
The number of bytes read.

GetPromiseValue - Parameters

valueType (ODValueType) - input
The type of the value from which the promise data is to be read.

offset (ODULong) - input
The offset from which the promise data is to be retrieve, from the beginning of the value.

length (ODULong) - input
The length, in bytes, of data to be retrieved.

value (ODByteArray *) - input
The byte array whose buffer contains the retrieved promise data.

sourcePart (ODPart **) - output
A reference to the part that made the promise.

rv (ODULong) - returns
The number of bytes read.

GetPromiseValue - Remarks

You call this method to read promise data without fulfilling the promise. This method first focuses the storage unit on the specified value of the focused property. It then starts reading data at the specified offset and stops after reading the number of bytes specified by the *length* parameter or after reaching the end of data in the focused value, whichever comes first.

When you call this method, the *_buffer* field of the *value* output parameter should be KODNULL; if it is not, the buffer to which that field points

will not be deallocated.

This method sets the *_buffer* field of the *value* output parameter to point to a memory block containing the promise data, the *_maximum* field to the specified length, and *_length* field to the number of bytes actually read.

This method sets the *sourcePart* parameter to a reference to the part that made the promise. This method does not increment the reference count of the part that made the promise.

When you no longer need the structure you pass the *value* parameter, you should deallocate that structure and its buffer.

GetPromiseValue - Exception Handling

kODErrInvalidValueType

The specified value type is improperly formed or illegal.

kODErrSUValueDoesNotExist

The focused property does not have a value with the specified value type.

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a property or a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetPromiseValue - Related Methods

Related Methods

- [ODStorageUnitView::IsPromiseValue](#)
- [ODStorageUnitView::SetPromiseValue](#)

GetPromiseValue - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetProperty

GetProperty - Syntax

This method returns the name of the property in the focus context.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODPropertyName    rv;

rv = GetProperty();
```

GetProperty Return Value - rv

rv ([ODPropertyName](#)) - returns
The name of the property in the focus context.

GetProperty - Parameters

rv ([ODPropertyName](#)) - returns
The name of the property in the focus context.

GetProperty - Remarks

When you no longer need the returned property name, you should deallocate it.

GetProperty - Exception Handling

KODErrInvalidProperty

The focus context of this storage-unit view is not a property or a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetProperty - Related Methods

Related Methods

- [ODStorageUnitView::AddProperty](#)
 - [ODStorageUnitView::Remove](#)
-

GetProperty - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetSize

GetSize - Syntax

This method returns the size of the data in the focus context.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetSize();
```

GetSize Return Value - rv

rv ([ODULong](#)) - returns

The size, in bytes, of the data in the focus context.

GetSize - Parameters

rv ([ODULong](#)) - returns

The size, in bytes, of the data in the focus context.

GetSize - Remarks

If the focus context is the storage unit that created this storage-unit view, this method returns the total size of all properties and values in the storage unit. If the focus context is a property, this method returns the total size of all values in the focused property. If the focus context is a value, this method returns the size of the data stream corresponding to the focused value.

If the focused value contains a promise value, the promise is fulfilled before the size of the value is evaluated.

GetSize - Exception Handling

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetSize - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

GetStorageUnit

GetStorageUnit - Syntax

This method returns a reference to the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnit      *rv;
```

```
rv = GetStorageUnit();
```

GetStorageUnit Return Value - rv

rv (ODStorageUnit *) - returns

A reference to the storage unit that created this storage-unit view.

GetStorageUnit - Parameters

rv (ODStorageUnit *) - returns
A reference to the storage unit that created this storage-unit view.

GetStorageUnit - Remarks

This method does not increment the reference count of the returned storage unit. For that reason, if you cache the returned storage unit, you should call its [Acquire](#) method to increment its reference count and then call its [Release](#) method when you are finished using it.

GetStorageUnit - Topics

Class:
ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetStrongStorageUnitRef

GetStrongStorageUnitRef - Syntax

This method creates a strong persistent reference to the specified storage unit.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    embeddedSUID;
ODStorageUnitRef   strongRef;

GetStrongStorageUnitRef(embeddedSUID, strongRef);
```

GetStrongStorageUnitRef Parameter - embeddedSUID

embeddedSUID ([ODStorageUnitID](#)) - input
The storage-unit ID of the storage unit whose persistent reference is desired.

GetStrongStorageUnitRef Parameter - strongRef

strongRef ([ODStorageUnitRef](#)) - output
The persistent reference to the storage unit specified by the *embeddedSUID* parameter.

GetStrongStorageUnitRef - Return Value

None.

GetStrongStorageUnitRef - Parameters

embeddedSUID ([ODStorageUnitID](#)) - input
The storage-unit ID of the storage unit whose persistent reference is desired.

strongRef ([ODStorageUnitRef](#)) - output
The persistent reference to the storage unit specified by the *embeddedSUID* parameter.

None.

GetStrongStorageUnitRef - Remarks

After this method executes successfully, you can call the [SetValue](#) method to store the resulting persistent reference, returned in the *strongRef* output parameter, into the focused value.

Important: The scope of a persistent reference is limited to the value in which it was created; therefore, when retrieving the storage unit, you must focus it on the same value that the persistent reference was created in.

For more information on persistent references, see the chapter on storage in the *OpenDoc Programming Guide* .

GetStrongStorageUnitRef - Exception Handling

kODerrIllegalNullStorageUnitInput

The *embeddedSUID* parameter is null.

kODerrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetStrongStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::GetWeakStorageUnitRef](#)
 - [ODStorageUnitView::IsStrongStorageUnitRef](#)
 - [ODStorageUnitView::SetValue](#)
-

GetStrongStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetType

GetType - Syntax

This method returns the type of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODValueType    rv;
```

```
rv = GetType();
```

GetType Return Value - rv

rv ([ODValueType](#)) - returns
The type of the focused value.

GetType - Parameters

rv ([ODValueType](#)) - returns
The type of the focused value.

GetType - Remarks

When you no longer need the returned value type, you should deallocate it.

GetType - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetType - Related Methods

Related Methods

- [ODStorageUnitView::SetType](#)
-

GetType - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

GetValue

GetValue - Syntax

This method reads data from the focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODULong      length;
ODByteArray  *value;
ODULong      rv;

rv = GetValue(length, value);
```

GetValue Parameter - length

length (ODULong) - input
The length, in bytes, of data to be read.

GetValue Parameter - value

value (ODByteArray *) - in/out
A byte array whose buffer is to contain the retrieved data.

GetValue Return Value - rv

rv (ODULong) - returns
The number of bytes read.

GetValue - Parameters

length (ODULong) - input
The length, in bytes, of data to be read.

value (ODByteArray *) - in/out
A byte array whose buffer is to contain the retrieved data.

rv (ODULong) - returns
The number of bytes read.

GetValue - Remarks

You call this method to read data from the focused value. If that value is a promise value, the promise is fulfilled before the data is read. This method starts reading data at the current offset and stops after reading the number of bytes specified by the *length* parameter or after reaching the end of the data in the currently focused value, whichever comes first.

When you call this method, the *_buffer* field of the *value* output parameter should be `kODNULL`; if it is not, the buffer to which that field points will not be deallocated.

This method sets the *_buffer* field of the *value* output parameter to point to a memory block containing the data that is read from the storage unit; it sets the *_maximum* field to the specified length and the *_length* field to the number of bytes actually read.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

GetValue - Exception Handling

`kODErrUnfocusedStorageUnit`

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetValue - Related Methods

Related Methods

- [ODStorageUnitView::GetSize](#)
 - [ODStorageUnitView::SetValue](#)
-

GetValue - Topics

Class:

`ODStorageUnitView`

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

GetWeakStorageUnitRef

GetWeakStorageUnitRef - Syntax

This method creates a weak persistent reference to the specified storage unit.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    embeddedSUID;
ODStorageUnitRef    weakRef;

GetWeakStorageUnitRef(embeddedSUID, weakRef);
```

GetWeakStorageUnitRef Parameter - embeddedSUID

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

GetWeakStorageUnitRef Parameter - weakRef

weakRef ([ODStorageUnitRef](#)) - output

The persistent reference to the storage unit specified by the *embeddedSUID* parameter.

GetWeakStorageUnitRef - Parameters

embeddedSUID ([ODStorageUnitID](#)) - input

The storage-unit ID of the storage unit whose persistent reference is desired.

weakRef ([ODStorageUnitRef](#)) - output

The persistent reference to the storage unit specified by the *embeddedSUID* parameter.

GetWeakStorageUnitRef - Remarks

After this method executes successfully, you can call the [SetValue](#) method to store the resulting persistent reference, returned in the *weakRef* output parameter, into the focused value.

Important: The scope of a persistent reference is limited to the value in which it was created; therefore, when retrieving the storage unit, you must focus it on the same value that the persistent reference was created in.

For more information on persistent references, see the chapter on storage in the *OpenDoc Programming Guide* .

GetWeakStorageUnitRef - Exception Handling

kODErrIllegalNullStorageUnitInput

The embeddedSUID parameter is null.

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

GetWeakStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::GetStrongStorageUnitRef](#)
- [ODStorageUnitView::IsWeakStorageUnitRef](#)
- [ODStorageUnitView::SetValue](#)

GetWeakStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IncrementGenerationNumber

IncrementGenerationNumber - Syntax

This method increments and returns the generation number of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODULong    rv;

rv = IncrementGenerationNumber();
```

IncrementGenerationNumber Return Value - rv

rv ([ODULong](#)) - returns
The generation number of the focused value.

IncrementGenerationNumber - Parameters

rv ([ODULong](#)) - returns
The generation number of the focused value.

IncrementGenerationNumber - Remarks

You can use the generation number of a value to tell whether the data in the value has changed. For example, when your part makes a significant change to the data in a value, you can call this method to increment its generation number.

IncrementGenerationNumber - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

IncrementGenerationNumber - Related Methods

Related Methods

- [ODStorageUnitView::GetGenerationNumber](#)
-

IncrementGenerationNumber - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

InsertValue

InsertValue - Syntax

This method inserts data into the focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODByteArray    *value;

InsertValue(value);
```

InsertValue Parameter - value

value (ODByteArray *) - input
A byte array whose buffer contains the data to be written.

InsertValue - Return Value

None.

InsertValue - Parameters

value (ODByteArray *) - input
A byte array whose buffer contains the data to be written.

None.

InsertValue - Remarks

You call this method to insert data into the focused value without overwriting the existing data at and beyond the current offset. If the focused value is currently a promise value, the promise is fulfilled before the data is written.

This method writes data to the focused value, starting at the current offset. If the focused value contained any data at and beyond the offset, that data appears after the inserted data. The size of the value is automatically increased to accommodate the inserted data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

InsertValue - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

InsertValue - Related Methods

Related Methods

- [ODStorageUnitView::DeleteValue](#)
- [ODStorageUnitView::SetValue](#)

InsertValue - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

Internalize

Internalize - Syntax

This method reads into memory all properties and values from the storage unit that created this storage-unit view.


```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitView      *rv;
```

```
rv = Internalize();
```

Internalize Return Value - rv

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Internalize - Parameters

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Internalize - Remarks

OpenDoc calls this method; your part does not call this method.

Internalize - Related Methods

Related Methods

- [ODStorageUnitView::Externalize](#)

Internalize - Topics

Class:

ODStorageUnitView

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

IsPromiseValue

IsPromiseValue - Syntax

This method returns indicates whether the focused value is a promise value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsPromiseValue();
```

IsPromiseValue Return Value - rv

rv (ODBoolean) - returns

A flag indicating whether the focused value is a promise value.

kODTrue	The focused value is a promise value.
kODFalse	The focused value is a regular value.

IsPromiseValue - Parameters

rv (ODBoolean) - returns

A flag indicating whether the focused value is a promise value.

kODTrue	The focused value is a promise value.
kODFalse	The focused value is a regular value.

IsPromiseValue - Remarks

If the focused value is a promise value, the promise is not resolved by this method.

IsPromiseValue - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

IsPromiseValue - Related Methods

Related Methods

- [ODStorageUnitView::GetPromiseValue](#)
- [ODStorageUnitView::SetPromiseValue](#)

IsPromiseValue - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

IsStrongStorageUnitRef

IsStrongStorageUnitRef - Syntax

This method indicates whether the specified reference is a strong persistent reference.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;
ODBoolean           rv;

rv = IsStrongStorageUnitRef(ref);
```

IsStrongStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input
The persistent reference to be tested. This value is assumed to be valid.

IsStrongStorageUnitRef Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the specified reference is a strong persistent reference.

kODTrue	The specified reference is a strong persistent reference.
kODFalse	The specified reference is not a strong persistent reference.

IsStrongStorageUnitRef - Parameters

ref ([ODStorageUnitRef](#)) - input
The persistent reference to be tested. This value is assumed to be valid.

rv ([ODBoolean](#)) - returns
A flag indicating whether the specified reference is a strong persistent reference.

kODTrue	The specified reference is a strong persistent reference.
kODFalse	The specified reference is not a strong persistent reference.

IsStrongStorageUnitRef - Remarks

Before calling this method, you can call the [IsValidStorageUnitRef](#) method to check whether the specified persistent reference is valid.

IsStrongStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit	The focus context of this storage-unit view is not a value.
----------------------------	---

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

IsStrongStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::GetStrongStorageUnitRef](#)
 - [ODStorageUnitView::IsValidStorageUnitRef](#)
 - [ODStorageUnitView::IsWeakStorageUnitRef](#)
-

IsStrongStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IsValidStorageUnitRef

IsValidStorageUnitRef - Syntax

This method indicates whether the specified persistent reference is valid.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;
ODBoolean           rv;

rv = IsValidStorageUnitRef(ref);
```

IsValidStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input

The persistent reference to be tested.

IsValidStorageUnitRef Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the specified persistent reference is valid.

kODTrue

The specified persistent reference is valid.

kODFalse

The specified persistent reference is not valid.

IsValidStorageUnitRef - Parameters

ref ([ODStorageUnitRef](#)) - input

The persistent reference to be tested.

rv ([ODBoolean](#)) - returns

A flag indicating whether the specified persistent reference is valid.

kODTrue

The specified persistent reference is valid.

kODFalse

The specified persistent reference is not valid.

IsValidStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

IsValidStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

IsWeakStorageUnitRef

IsWeakStorageUnitRef - Syntax

This method indicates whether the specified reference is a weak persistent reference.

```

#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef    ref;
ODBoolean           rv;

rv = IsWeakStorageUnitRef(ref);

```

IsWeakStorageUnitRef Parameter - ref

ref ([ODStorageUnitRef](#)) - input
 The persistent reference to be tested. This value is assumed to be valid.

IsWeakStorageUnitRef Return Value - rv

rv ([ODBoolean](#)) - returns
 A flag indicating whether specified reference is a weak persistent reference.

kODTrue	The specified reference is a weak persistent reference.
kODFalse	The specified reference is not a weak persistent reference.

IsWeakStorageUnitRef - Parameters

ref ([ODStorageUnitRef](#)) - input
 The persistent reference to be tested. This value is assumed to be valid.

rv ([ODBoolean](#)) - returns
 A flag indicating whether specified reference is a weak persistent reference.

kODTrue	The specified reference is a weak persistent reference.
kODFalse	The specified reference is not a weak persistent reference.

IsWeakStorageUnitRef - Remarks

Before calling this method, you can call the [IsValidStorageUnitRef](#) method to check whether the specified persistent reference is valid.

IsWeakStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

IsWeakStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::GetWeakStorageUnitRef](#)
- [ODStorageUnitView::IsStrongStorageUnitRef](#)
- [ODStorageUnitView::IsValidStorageUnitRef](#)

IsWeakStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Remove

Remove - Syntax

This method removes all properties and values in the focus context from the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *rv;

rv = Remove();
```

Remove Return Value - rv

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Remove - Parameters

rv (ODStorageUnitView *) - returns
A reference to this storage-unit view.

Remove - Remarks

If the focus context of this storage-unit view is the entire storage unit, this method removes all properties and their values. If the focus context is a property, this method removes the focused property and all its values. If the focus context is a value, this method removes the focused value.

After this method executes successfully, the storage unit that created this storage-unit view is unfocused. The focus context of this storage-unit view is unchanged.

This method should be used with caution; if it executes successfully, it makes the focus context of this storage-unit view invalid.

Remove - Exception Handling

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

Remove - Related Methods

Related Methods

- [ODStorageUnitView::AddProperty](#)
 - [ODStorageUnitView::AddValue](#)
-

Remove - Topics

Class:
ODStorageUnitView

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

RemoveStorageUnitRef

RemoveStorageUnitRef - Syntax

This method makes a persistent reference invalid in the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitRef      aRef;
ODStorageUnitView     *rv;

rv = RemoveStorageUnitRef(aRef);
```

RemoveStorageUnitRef Parameter - aRef

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be removed.

RemoveStorageUnitRef Return Value - rv

rv ([ODStorageUnitView *](#)) - returns
A reference to this storage-unit view.

RemoveStorageUnitRef - Parameters

aRef ([ODStorageUnitRef](#)) - input
The persistent reference to be removed.

rv ([ODStorageUnitView *](#)) - returns
A reference to this storage-unit view.

RemoveStorageUnitRef - Remarks

This method does not change the data in the focused value, but after this method is called, the specified persistent reference is no longer valid. To remove data corresponding to the persistent reference, you must call the [DeleteValue](#) method.

RemoveStorageUnitRef - Exception Handling

kODErrUnfocusedStorageUnit

The current focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

RemoveStorageUnitRef - Related Methods

Related Methods

- [ODStorageUnitView::DeleteValue](#)
-

RemoveStorageUnitRef - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetName

SetName - Syntax

This method sets the name of the storage unit that created this storage-unit view.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnitName    name;
```

```
SetName (name);
```

SetName Parameter - name

name ([ODStorageUnitName](#)) - input

The name to be assigned to the storage unit that created this storage-unit view.

SetName - Return Value

None.

SetName - Parameters

name ([ODStorageUnitName](#)) - input

The name to be assigned to the storage unit that created this storage-unit view.

None.

SetName - Related Methods

Related Methods

- [ODStorageUnitView::GetName](#)
-

SetName - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

SetOffset

SetOffset - Syntax

This method sets the offset of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>

ODULong    offset;

SetOffset(offset);
```

SetOffset Parameter - offset

offset ([ODULong](#)) - input

The new offset, in bytes, of the read/write position from the beginning of the data stream in the focused value.

SetOffset - Return Value

None.

SetOffset - Parameters

offset ([ODULong](#)) - input

The new offset, in bytes, of the read/write position from the beginning of the data stream in the focused value.

None.

SetOffset - Remarks

You can call this method if you want to read or write data at a particular position in the focused value. An offset of 0 means the beginning of the data stream corresponding to the focused value; an offset equal to the current size of the focused value (as returned by the [GetSize](#) method) means the end of the data stream. You may not specify an offset larger than the current size of the focused value.

SetOffset - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

SetOffset - Related Methods

Related Methods

- [ODStorageUnitView::GetOffset](#)
 - [ODStorageUnitView::GetSize](#)
-

SetOffset - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetPromiseValue

SetPromiseValue - Syntax

This method writes data to the specified value of the focused property, creating the value if it does not exist and making the value a promise value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODValueType      valueType;
ODULong          offset;
ODByteArray      *value;
ODPart           *sourcePart;
```

```
SetPromiseValue(valueType, offset, value,
                 sourcePart);
```

SetPromiseValue Parameter - valueType

valueType ([ODValueType](#)) - input
The type of the value where the promise data is to be written.

SetPromiseValue Parameter - offset

offset ([ODULong](#)) - input
The offset from which the promise data is to be stored, from the beginning of the value.

SetPromiseValue Parameter - value

value ([ODByteArray *](#)) - input
A byte array whose buffer contains the promise data to be written.

SetPromiseValue Parameter - sourcePart

sourcePart ([ODPart *](#)) - input
A reference to the part that made the promise.

SetPromiseValue - Return Value

None.

SetPromiseValue - Parameters

valueType ([ODValueType](#)) - input
The type of the value where the promise data is to be written.

offset ([ODULong](#)) - input
The offset from which the promise data is to be stored, from the beginning of the value.

value ([ODByteArray *](#)) - input
A byte array whose buffer contains the promise data to be written.

sourcePart ([ODPart *](#)) - input
A reference to the part that made the promise.

None.

SetPromiseValue - Remarks

You call this method to write a promise for a value of the specified type in the focused property. You may call this method multiple times to promise values of different types or to write to different offsets in the same value.

This method writes data to the specified value, starting at the specified offset (inclusive), and overwrites any data at and beyond the offset. If the current offset plus the length of data being written is greater than the current size of the value (as returned by the [GetSize](#) method), the size of the value is automatically increased to accommodate the new data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

SetPromiseValue - Exception Handling

[kODErrInvalidType](#) The specified value type improperly formed or illegal.

[kODErrUnfocusedStorageUnit](#) The focus context of this storage-unit view is not a property or a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

SetPromiseValue - Related Methods

Related Methods

- [ODStorageUnitView::GetPromiseValue](#)
- [ODStorageUnitView::GetSize](#)
- [ODStorageUnitView::IsPromiseValue](#)

SetPromiseValue - Topics

Class:
[ODStorageUnitView](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

SetType

SetType - Syntax

This method sets the type of the focused value.

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODValueType    valueType;
```

```
SetType(valueType);
```

SetType Parameter - valueType

valueType ([ODValueType](#)) - input
The new type of the focused value.

SetType - Return Value

None.

SetType - Parameters

valueType ([ODValueType](#)) - input
The new type of the focused value.

None.

SetType - Remarks

This method should be used with caution; it may make this storage-unit view invalid.

SetType - Exception Handling

kODErrInvalidValueType

The specified value type is improperly formed or illegal.

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

SetType - Related Methods

Related Methods

- [ODStorageUnitView::GetType](#)
-

SetType - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

SetValue

SetValue - Syntax

This method writes data to the focused value, starting at the offset (inclusive).

```
#define INCL_ODSTORAGEUNITVIEW
#define INCL_ODAPI
#include <os2.h>
```

```
ODByteArray      *value;
```

```
SetValue(value);
```

SetValue Parameter - value

value ([ODByteArray *](#)) - input
A byte array whose buffer contains the data to be written.

SetValue - Return Value

None.

SetValue - Parameters

value ([ODByteArray *](#)) - input
A byte array whose buffer contains the data to be written.

None.

SetValue - Remarks

You call this method to write data to the focused value. If that value is currently a promise value, the promise is fulfilled before the data is written.

This method writes data to the focused value, starting at the current offset, and overwrites any data at and beyond the offset. If the current offset plus the length of data being written is greater than the current size of the value (as returned by the [GetSize](#) method), the size of the value is automatically increased to accommodate the new data.

When you no longer need the structure you pass as the *value* parameter, you should deallocate that structure and its buffer.

SetValue - Exception Handling

kODErrUnfocusedStorageUnit

The focus context of this storage-unit view is not a value.

If the storage-unit cursor for this storage-unit view does not represent a legal focus context for the storage unit that created this storage-unit view, this method throws exceptions raised by the [FocusWithCursor](#) method of that storage unit.

SetValue - Related Methods

Related Methods

- [ODStorageUnitView::GetSize](#)
- [ODStorageUnitView::GetValue](#)

SetValue - Topics

Class:

[ODStorageUnitView](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

ODTemplates

Class Definition File: ODTEMPS.IDL

Class Hierarchy

SOMObject

ODObject

ODTemplates

Description

Templates in OpenDoc behave like templates in the Workplace Shell. They can be dragged onto the document shell and inserted into the current document and onto the open desktop. When a template object is dropped, it starts the document shell with the template part as the root part of the document.

When a user installs OpenDoc for the first time, each part handler that is registered with the part registry causes the creation of a template in the OpenDoc Templates folder. Templates can also be created at run time by calling the [CreateTemplate](#) method.

At installation, a Workplace Shell folder is created with the title "OpenDoc Templates" and with an object ID of "<OD_TEMPS>". This object ID is a constant within the ODTemplates class. On each creation of a template, the calling method does not need to specify the location of the template being created when using this constant; however, the part creating a template must specify a unique object ID for each template it creates with the format "<OD_myobjid>" giving each template a unique Workplace Shell object ID. This enables templates to be removed from the folder, if necessary, and allows for easy discernment between "true" Workplace Shell objects and OpenDoc objects on the desktop.

Methods

The methods defined by the ODTemplates class include:

- [CreateTemplate](#)
- [DeleteTemplate](#)
- [GetStorageUnit](#)

Overridden Methods

There are currently no methods overridden by the ODTemplates class.

CreateTemplate (OS/2)

CreateTemplate (OS/2) - Syntax

This method creates a template of the specified part with the content of the part's storage unit in the data file.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;
ODSession   *session;
string      ObjectID;
string      ObjectTitle;
ODStorageUnit *unit;
ODBoolean   rv;

rv = CreateTemplate(part, session, ObjectID,
                   ObjectTitle, unit);
```

CreateTemplate (OS/2) Parameter - part

part (ODPart *) - input
A pointer to the part object of the calling part.

CreateTemplate (OS/2) Parameter - session

session (ODSession *) - input
A pointer to the session object of the calling part.

CreateTemplate (OS/2) Parameter - ObjectID

ObjectID (string) - input
A Workplace Shell object ID of the format "<OD_myobjid>" or kODNULL if the object ID is generated using the part kind name.

CreateTemplate (OS/2) Parameter - ObjectTitle

ObjectTitle (string) - input
A string to be used as the template's title.

CreateTemplate (OS/2) Parameter - unit

unit (ODStorageUnit *) - input
A pointer to the part's storage unit.

CreateTemplate (OS/2) Return Value - rv

rv (ODBoolean) - returns
A flag indicating whether a template was created successfully.

kODTrue	The template was created.
kODFalse	The template was not created.

CreateTemplate (OS/2) - Parameters

part (ODPart *) - input
A pointer to the part object of the calling part.

session (ODSession *) - input
A pointer to the session object of the calling part.

ObjectID (string) - input
A Workplace Shell object ID of the format "<OD_myobjid>" or kODNULL if the object ID is generated using the part kind name.

ObjectTitle (string) - input
A string to be used as the template's title.

unit (ODStorageUnit *) - input
A pointer to the part's storage unit.

rv (ODBoolean) - returns
A flag indicating whether a template was created successfully.

kODTrue	The template was created.
kODFalse	The template was not created.

CreateTemplate (OS/2) - Remarks

This template is placed in the OpenDoc Templates folder on the desktop with the specified object ID as its Workplace Shell object ID.

CreateTemplate (OS/2) - Exception Handling

kODErrFocusNotRegistered

The requested focus is not registered.

kODErrOutOfMemory

There is not enough memory to allocate the focus-owner iterator object.

CreateTemplate (OS/2) - Related Methods

Related Methods

- [ODArbitrator::AcquireFocusOwner](#)
- [ODSession::Tokenize](#)

CreateTemplate (OS/2) - Topics

Class:

ODTemplates

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

DeleteTemplate (OS/2)

DeleteTemplate (OS/2) - Syntax

This method deletes a template with the specified object ID.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

string      ObjectID;
ODBoolean   rv;

rv = DeleteTemplate(ObjectID);
```

DeleteTemplate (OS/2) Parameter - ObjectID

ObjectID ([string](#)) - input
The Workplace Shell object ID of the template to be deleted.

DeleteTemplate (OS/2) Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether a template was deleted successfully.

kODTrue	The template was deleted.
kODFalse	The template was not deleted.

DeleteTemplate (OS/2) - Parameters

ObjectID ([string](#)) - input
The Workplace Shell object ID of the template to be deleted.

rv ([ODBoolean](#)) - returns
A flag indicating whether a template was deleted successfully.

kODTrue	The template was deleted.
kODFalse	The template was not deleted.

DeleteTemplate (OS/2) - Topics

Class:
ODTemplates

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetStorageUnit (OS/2)

GetStorageUnit (OS/2) - Syntax

This method returns an empty storage unit into which the part can clone its storage unit.


```
#define INCL_ODTEMPLATES
#define INCL_ODAPI
#include <os2.h>

ODSession      *session;
ODStorageUnit  *rv;

rv = GetStorageUnit(session);
```

GetStorageUnit (OS/2) Parameter - session

session (ODSession *) - input
A pointer to the session object of the calling part.

GetStorageUnit (OS/2) Return Value - rv

rv (ODStorageUnit *) - returns
A pointer to the new fully-operational storage unit.

GetStorageUnit (OS/2) - Parameters

session (ODSession *) - input
A pointer to the session object of the calling part.

rv (ODStorageUnit *) - returns
A pointer to the new fully-operational storage unit.

GetStorageUnit (OS/2) - Topics

Class:
ODTemplates

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODTransform

Class Definition File: TRNSFORM.IDL

Class Hierarchy

SOMObject
 ODObject
 ODBaseTransform
 ODTransform

Description

An object of the ODTransform class maps points and shapes from one coordinate system to another. This can be viewed as modifying a shape, for example, by scaling or rotating it.

A transform uses a 3-by-3 matrix to perform two-dimensional transformations. The simplest transform is an identity transform, which has no effect on any points or shapes that it transforms.

Your part creates a new identity transform by calling the [CreateTransform](#) method of a frame, the [CreateTransform](#) method of facet, or the [NewTransform](#) method of an existing transform. Your part can create a copy of an existing transform by calling that transform's [Copy](#) method.

You can use a transform to perform the following transformations on coordinates or shapes:

Perspective	Modifies the positions of points to give a three-dimensional effect.
Rotation	Changes the angle of rotation of a shape, rotating all points around a given point.
Scaling	Changes the size of a shape.
Skewing	Changes the slant applied to a shape.
Translation (offset)	Shifts the position of a shape.

For more information on matrices and transformations in two-dimensional drawing, you can consult any standard computer-graphics textbook. You can also refer to the chapter on drawing in the *OpenDoc Programming Guide*.

You do not need to subclass [ODTransform](#); however, you can provide for new transforms by creating subclasses of [ODTransform](#). For example, you can define new transforms that do not use transformation matrices. These new transforms might perform more complex operations, such as morphing or wrapping around a 3D surface.

Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODTransform.

somInit

This method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODTransform, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in this transform object. The SOM library calls this method when this transform object is created. You must not do anything that might cause this method to fail. This limits you to operations like setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this transform object's subclass-specific initialization method; see also the method [InitTransform](#).

somUninit

This method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODTransform, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for this transform object, including any storage related to additional instance variables initialized in this transform object. The SOM library calls this method when this transform object is deleted; this method must not fail.

Purge

This method frees memory on request; it is inherited from the [ODObject](#) class.

```
ODSize Purge (ODSize size);
```

If you subclass [ODObject](#), you can override this method and should do so if it creates caches and temporary buffers. If you subclass ODTransform, you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method because you

will need it to compute the value to return from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the sum as the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

Methods

The methods defined by the ODTransform class include:

Creating Transforms

- [Copy](#)
- [NewTransform](#)

Applying Transforms

- [InvertPoint](#)
- [InvertShape](#)
- [TransformPoint](#)
- [TransformPoints](#)
- [TransformShape](#)

Testing Transforms

- [GetType](#)
- [HasMatrix](#)
- [IsSameAs](#)

Manipulating Matrices

- [CopyFrom](#)
- [GetMATRIXLF](#)
- [GetMatrix](#)
- [Invert](#)
- [PostCompose](#)
- [PreCompose](#)
- [ReadFrom](#)
- [Reset](#)
- [SetMatrix](#)
- [SetMATRIXLF](#)
- [WriteTo](#)

Manipulating Translation Values

- [GetOffset](#)
- [GetPreScaleOffset](#)
- [MoveBy](#)
- [SetOffset](#)

Manipulating Scaling Factors

- [GetScale](#)
- [ScaleBy](#)
- [ScaleDownBy](#)

Initializing

- [InitTransform](#)

Overridden Methods

There are currently no methods overridden by the ODTransform class.

Copy

Copy - Syntax

This method creates a new transform that is a copy of this transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = Copy();
```

Copy Return Value - rv

rv (ODTransform *) - returns
A reference to the newly created transform.

Copy - Parameters

rv (ODTransform *) - returns
A reference to the newly created transform.

Copy - Remarks

The new transform does not share any data with this transform; therefore, you can modify each of the transforms independently.

This method initializes the reference count of the returned transform. When you have finished using that transform, you should call its [Release](#) method.

Copy - Exception Handling

kODErrOutOfMemory

There is not enough memory to copy the transform.

Copy - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

Copy - Topics

Class:ODTransform

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Override Policy](#)[Exception Handling](#)

CopyFrom

CopyFrom - Syntax

This method modifies this transform to make it equivalent to the specified source transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *sourceTransform;
ODTransform      *rv;

rv = CopyFrom(sourceTransform);
```

CopyFrom Parameter - sourceTransform

sourceTransform (ODTransform *) - input

A reference to the source transform to be copied.

CopyFrom Return Value - rv

rv (ODTransform *) - returns

A reference to this transform modified to be a copy of the specified transform.

CopyFrom - Parameters

sourceTransform (ODTransform *) - input
A reference to the source transform to be copied.

rv (ODTransform *) - returns
A reference to this transform modified to be a copy of the specified transform.

CopyFrom - Remarks

After this method executes successfully, this transform and the source transform do not share data; therefore, you can modify each of the transforms independently.

CopyFrom - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

CopyFrom - Topics

Class:
ODTransform

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Override Policy](#)

GetMatrix

GetMatrix - Syntax

This method copies this transform's matrix into the specified structure.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODMatrix      *matrix;

GetMatrix(matrix);
```

GetMatrix Parameter - matrix

matrix (ODMatrix *) - output

The structure in which the matrix is to be returned.

GetMatrix - Return Value

None.

GetMatrix - Parameters

matrix (ODMatrix *) - output

The structure in which the matrix is to be returned.

None.

GetMatrix - Remarks

If you use transforms of the class [ODTransform](#) and you have also created a subclass of [ODTransform](#) that applies complex transformation effects that cannot be represented by matrixes, your part needs to check whether a given transform has a matrix before calling this method. To do so, you should call the transform's [HasMatrix](#) method; only if that method returns kODTrue should you call the transform's [GetMatrix](#) method.

GetMatrix - Exception Handling

kODErrTransformErr

The tranform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

GetMatrix - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method must not call its inherited method. Instead, it should raise an kODErrTransformErr exception. You can detect that a transform of your class has no matrix by calling its [HasMatrix](#) method.

GetMatrix - Related Methods

Related Methods

- [ODTransform::HasMatrix](#)
- [ODTransform::SetMatrix](#)

GetMatrix - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

GetMATRIXLF (OS/2)

GetMATRIXLF (OS/2) - Syntax

This method returns a GPI matrix based on the OpenDoc matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
MATRIXLF      *mtx;
```

```
GetMATRIXLF (mtx) ;
```

GetMATRIXLF (OS/2) Parameter - mtx

mtx ([MATRIXLF *](#)) - output

The GPI matrix-elements structure.

GetMATRIXLF (OS/2) - Return Value

None.

GetMATRIXLF (OS/2) - Parameters

mtx ([MATRIXLF *](#)) - output
The GPI matrix-elements structure.

None.

GetMATRIXLF (OS/2) - Remarks

Because GPI matrices do not support all transformations which may be represented by an OpenDoc transformation matrix, these two matrices can not represent the same transform.

GetMATRIXLF (OS/2) - Topics

Class:
ODTransform

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetOffset

GetOffset - Syntax

This method returns this transform's translation values (also called offset values) in the specified structure.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODPoint    *offset;
```

```
GetOffset(offset);
```

GetOffset Parameter - offset

offset ([ODPoint *](#)) - output

A point specifying the horizontal (x) and vertical (y) translation values of this transform.

GetOffset - Return Value

None.

GetOffset - Parameters

offset ([ODPoint *](#)) - output

A point specifying the horizontal (x) and vertical (y) translation values of this transform.

None.

GetOffset - Remarks

The returned horizontal and vertical translation values are located in this transform's matrix; they are the first two elements of the bottom row in the matrix.

GetOffset - Related Methods

Related Methods

- [ODTransform::GetPreScaleOffset](#)
 - [ODTransform::SetOffset](#)
-

GetOffset - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

GetPreScaleOffset

GetPreScaleOffset - Syntax

This method returns, in the specified structure, the offset to use if you are going to apply the offset before scaling.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *offset;

GetPreScaleOffset(offset);
```

GetPreScaleOffset Parameter - offset

offset ([ODPoint *](#)) - output
A point specifying the horizontal (x) and vertical (y) translation values to use before scaling.

GetPreScaleOffset - Return Value

None.

GetPreScaleOffset - Parameters

offset ([ODPoint *](#)) - output
A point specifying the horizontal (x) and vertical (y) translation values to use before scaling.

None.

GetPreScaleOffset - Remarks

This method is useful if you want to transform your data by first offsetting and then scaling it. It should not be used if the transformation involves rotation, skewing, or perspective.

GetPreScaleOffset - Related Methods

Related Methods

- [ODTransform::GetOffset](#)
-

GetPreScaleOffset - Topics

Class:

ODTransform

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

GetScale

GetScale - Syntax

This method returns this transform's horizontal and vertical scaling factors in the specified structure.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *scale;

GetScale(scale);
```

GetScale Parameter - scale

scale ([ODPoint *](#)) - output
A point specifying the horizontal (x) and vertical (y) scaling factors.

GetScale - Return Value

None.

GetScale - Parameters

scale ([ODPoint *](#)) - output
A point specifying the horizontal (x) and vertical (y) scaling factors.

None.

GetScale - Remarks

Two elements in the transform's matrix specify the amount by which a shape is scaled. The horizontal scaling factor is determined by the element in the top-left corner of the matrix. The vertical scaling factor is governed by element in the center of the matrix.

GetScale - Topics

Class:
ODTransform

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetType

GetType - Syntax

This method returns the transform type of this transform.

```

#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransformType    rv;

rv = GetType();

```

GetType Return Value - rv

rv ([ODTransformType](#)) - returns

The transform type of this transform. This parameter can be set to one of the following values:

kODIdentityXform	Identity transform.
kODLinearTranslateXform	Scale, rotate, skew, and translation (offset).
kODLinearXform	Scale, rotate and skew.
kODPerspectiveXform	Perspective transformation, which applies to 3D or distortion effects.
kODScaleTranslateXform	Scale and translation (offset).
kODScaleXform	Pure scale.
kODTranslateXform	Pure translation (offset).

GetType - Parameters

rv ([ODTransformType](#)) - returns

The transform type of this transform. This parameter can be set to one of the following values:

kODIdentityXform	Identity transform.
kODLinearTranslateXform	Scale, rotate, skew, and translation (offset).
kODLinearXform	Scale, rotate and skew.
kODPerspectiveXform	Perspective transformation, which applies to 3D or distortion effects.
kODScaleTranslateXform	Scale and translation (offset).
kODScaleXform	Pure scale.
kODTranslateXform	Pure translation (offset).

GetType - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

HasMatrix

HasMatrix - Syntax

This method should indicate whether this transform uses a matrix to describe its transformation.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = HasMatrix();
```

HasMatrix Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the transform object uses a matrix.

kODTrue

The transform object uses a matrix.

kODFalse

The transform object does not use a matrix.

HasMatrix - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether the transform object uses a matrix.

kODTrue

The transform object uses a matrix.

kODFalse

The transform object does not use a matrix.

HasMatrix - Remarks

Every objects of the [ODTransform](#) class uses a matrix; hence, this method returns kODTrue. However, if you use transforms of the

[ODTransform](#) class and you also have created a subclass of [ODTransform](#) that applies complex transformation effects that cannot be represented by matrices, you can call this method to test whether a particular transform belongs to a class that uses a transform matrix.

HasMatrix - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method must not call its inherited method; instead it should return `kODFalse`.

HasMatrix - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

InitTransform

InitTransform - Syntax

This method initializes this transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
InitTransform();
```

InitTransform - Return Value

None.

InitTransform - Parameters

None.

InitTransform - Remarks

This method is not called directly to initialize this transform object, but is called by a subclass-specific initialization method. By convention, every subclass of ODTransform should have a separate initialization method (for example, the InitMyTransform method) that is called when an instance of that subclass is created. The initialization method may have additional parameters beyond those of the InitTransform method. The InitMyTransform method should call the inherited InitTransform method at the beginning of its implementation.

If you subclass [ODTransform](#), your subclass-specified initialization method, rather than its somlInit method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your transform.

InitTransform - Override Policy

If you subclass [ODTransform](#), you should not override this method.

InitTransform - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

Invert

Invert - Syntax

This method should invert this transform's matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODTransform      *rv;
```

```
rv = Invert();
```

Invert Return Value - rv

rv (ODTransform *) - returns

A reference to this transform with its matrix changed to represent the inverse of its original matrix.

Invert - Parameters

rv (ODTransform *) - returns

A reference to this transform with its matrix changed to represent the inverse of its original matrix.

Invert - Remarks

The inverse of a transform is the mathematical inverse of its matrix. It has the exact opposite geometric effect of the original transform.

Invert - Exception Handling

kODErrOutOfMemory

There is not enough memory to invert the matrix.

kODErrTransformErr

The transform has no inverse. This is not true of most real-world transformations, only for those that perform transformations such as flattening a shape into a single line or point.

Invert - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

Invert - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InvertPoint

InvertPoint - Syntax

This method should modify the location of the specified point by applying the inverse of this transform matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;

InvertPoint (point);
```

InvertPoint Parameter - point

point (ODPoint *) - in/out

The point to be modified. On return, the fields of this structure have been modified to represent the modified point.

InvertPoint - Return Value

None.

InvertPoint - Parameters

point (ODPoint *) - in/out

The point to be modified. On return, the fields of this structure have been modified to represent the modified point.

None.

InvertPoint - Exception Handling

kODErrOutOfMemory

There is not enough memory to compute the inverse matrix.

kODErrTransformErr

The tranform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

InvertPoint - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

InvertPoint - Related Methods

Related Methods

- [ODTransform::TransformPoint](#)

InvertPoint - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

InvertShape

InvertShape - Syntax

This method should modify the specified shape by applying the inverse of this transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *shape;
```

```
InvertShape(shape);
```

InvertShape Parameter - shape

shape (ODShape *) - input

A reference to the shape object whose geometric representation is to be modified by the inverse of this transform.

InvertShape - Return Value

None.

InvertShape - Parameters

shape (ODShape *) - input

A reference to the shape object whose geometric representation is to be modified by the inverse of this transform.

None.

InvertShape - Remarks

This method is operationally equivalent to the [InverseTransform](#) method of the specified shape.

InvertShape - Exception Handling

kODErrOutOfMemory

There is not enough memory to invert the shape.

kODErrTransformErr

The tranform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

InvertShape - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

InvertShape - Related Methods

Related Methods

- [ODShape::InverseTransform](#)
-

InvertShape - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Override Policy](#)
[Exception Handling](#)
[Related Methods](#)

IsSameAs

IsSameAs - Syntax

This method should indicate whether this transform is identical to the specified transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *compareTransform;
ODBoolean        rv;

rv = IsSameAs(compareTransform);
```

IsSameAs Parameter - compareTransform

compareTransform (ODTransform *) - input

A reference to the transform to be used for comparison.

IsSameAs Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the transforms are equivalent (that is, they describe the same transformation).

kODTrue

The transforms are equivalent.

kODFalse

The transforms are different.

IsSameAs - Parameters

compareTransform (ODTransform *) - input

A reference to the transform to be used for comparison.

rv ([ODBoolean](#)) - returns

A flag indicating whether the transforms are equivalent (that is, they describe the same transformation).

kODTrue

The transforms are equivalent.

kODFalse

The transforms are different.

IsSameAs - Remarks

The transform objects are equal if, and only if, their matrices are identical (within a rounding error defined to be 7/32 768 units of a fixed integer representation) or if one is an exact multiple of the other.

IsSameAs - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

IsSameAs - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

MoveBy

MoveBy - Syntax

This method offsets this transform's horizontal and vertical translation values by the specified amount.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;
ODTransform  *rv;

rv = MoveBy(point);
```

MoveBy Parameter - point

point (ODPoint *) - input
A point specifying the horizontal (x) and vertical (y) translations to be added to the current translation values.

MoveBy Return Value - rv

rv (ODTransform *) - returns
A reference to this transform after the offset operation.

MoveBy - Parameters

point (ODPoint *) - input
A point specifying the horizontal (x) and vertical (y) translations to be added to the current translation values.

rv (ODTransform *) - returns
A reference to this transform after the offset operation.

MoveBy - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

NewTransform

NewTransform - Syntax

This method creates a new identity transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = NewTransform();
```

NewTransform Return Value - rv

rv (ODTransform *) - returns

A reference to the newly created transform or KODNULL if an error occurred.

NewTransform - Parameters

rv (ODTransform *) - returns

A reference to the newly created transform or KODNULL if an error occurred.

NewTransform - Remarks

This method initializes the reference count of the returned transform. When you have finished using that transform, you should call its [Release](#) method.

NewTransform - Exception Handling

kODErrOutOfMemory

There is not enough memory to create a new transform.

NewTransform - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

PostCompose

PostCompose - Syntax

This method should modify this transform's matrix by postmultipliing it with the specified transform's matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *ttransform;
ODTransform      *rv;

rv = PostCompose(transform);
```

PostCompose Parameter - transform

transform (ODTransform *) - input

A reference to the transform whose matrix is to be postmultiplied with this transform's matrix.

PostCompose Return Value - rv

rv (ODTransform *) - returns

A reference to this transform after the postcompose operation.

PostCompose - Parameters

transform (ODTransform *) - input

A reference to the transform whose matrix is to be postmultiplied with this transform's matrix.

rv (ODTransform *) - returns

A reference to this transform after the postcompose operation.

PostCompose - Remarks

Postcomposing multiplies this transform's matrix on the right side by the specified transform:

```
this <- this x transform
```

The resulting transform has the same effect as applying the two original transforms in sequence: first this transform, then the other.

PostCompose - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

PostCompose - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

PreCompose

PreCompose - Syntax

This method should modify this transform's matrix by premultiplying it with the specified transform's matrix

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *ttransform;
ODTransform      *rv;

rv = PreCompose(ttransform);
```

PreCompose Parameter - transform

transform (ODTransform *) - input
A reference to the transform whose matrix is to be premultiplied with this transform's matrix.

PreCompose Return Value - rv

rv (ODTransform *) - returns
A reference to this transform after the precompose operation.

PreCompose - Parameters

transform (ODTransform *) - input
A reference to the transform whose matrix is to be premultiplied with this transform's matrix.

rv (ODTransform *) - returns
A reference to this transform after the precompose operation.

PreCompose - Remarks

Precomposing multiplies this transform's matrix on the left side by the specified transform.

```
this <- transform x this
```

The resulting transform has the same effect as applying the two original transforms in sequence: first the other transform, then this one.

PreCompose - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

PreCompose - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

ReadFrom

ReadFrom - Syntax

This method should read this transform's matrix from the specified storage unit.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;

ReadFrom(storageUnit);
```

ReadFrom Parameter - storageUnit

storageUnit (ODStorageUnit *) - input

A reference to the storage unit from which the matrix is to be read.

ReadFrom - Return Value

None.

ReadFrom - Parameters

storageUnit (ODStorageUnit *) - input

A reference to the storage unit from which the matrix is to be read.

None.

ReadFrom - Remarks

Before calling this method, you must focus the storage unit on the property from which the matrix elements are to be written. The matrix is read from the value of type kODTransform in the focused property. If no such value exists, this transform is reset to an identity transform.

ReadFrom - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or a value.

ReadFrom - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

ReadFrom - Related Methods

Related Methods

- [ODTransform::WriteTo](#)
-

ReadFrom - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Override Policy](#)

Reset

Reset - Syntax

This method should change this transform to the identity transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = Reset();
```

Reset Return Value - rv

rv (ODTransform *) - returns
A reference to this transform reset to the identity transform.

Reset - Parameters

rv (ODTransform *) - returns
A reference to this transform reset to the identity transform.

Reset - Remarks

Except for transforms created by the [Copy](#) method, newly created transforms start out as identity transforms; this method changes a transform back to the initial state.

Reset - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited

method at any point in your implementation (it does not matter where).

Reset - Topics

Class:

ODTransform

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

ScaleBy

ScaleBy - Syntax

This method scales this transform by the specified vertical and horizontal scaling factors.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *scale;
ODTransform  *rv;

rv = ScaleBy(scale);
```

ScaleBy Parameter - scale

scale ([ODPoint *](#)) - input

A point specifying the horizontal (x) and vertical (y) scaling factors.

ScaleBy Return Value - rv

rv ([ODTransform *](#)) - returns

A reference to this transform after the scaling operation.

ScaleBy - Parameters

scale ([ODPoint](#) *) - input
A point specifying the horizontal (x) and vertical (y) scaling factors.

rv (ODTransform *) - returns
A reference to this transform after the scaling operation.

ScaleBy - Exception Handling

kODErrTransformErr

The transform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

ScaleBy - Related Methods

Related Methods

- `ODTransform::ScaleDownBy`

ScaleBy - Topics

Class:

ODTransform

Select an item:

Syntax

Parameters

Returns

Exception Handling

Related Methods

ScaleDownBy

ScaleDownBy - Syntax

This method scales this transform by the reciprocal of the specified vertical and horizontal scaling factors.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *scale;
ODTransform  *rv;

rv = ScaleDownBy(scale);
```

ScaleDownBy Parameter - scale

scale ([ODPoint *](#)) - input
A point specifying the horizontal (x) and vertical (y) scaling factors.

ScaleDownBy Return Value - rv

rv ([ODTransform *](#)) - returns
A reference to this transform after the scaling operation.

ScaleDownBy - Parameters

scale ([ODPoint *](#)) - input
A point specifying the horizontal (x) and vertical (y) scaling factors.

rv ([ODTransform *](#)) - returns
A reference to this transform after the scaling operation.

ScaleDownBy - Remarks

This method is the inverse of the transform's [ScaleBy](#) method.

ScaleDownBy - Related Methods

Related Methods

- [ODTransform::ScaleBy](#)

ScaleDownBy - Topics

Class:ODTransform

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

SetMatrix

SetMatrix - Syntax

This method replaces this transform's matrix with the specified matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODMatrix      *matrix;
ODTransform    *rv;
```

```
rv = SetMatrix(matrix);
```

SetMatrix Parameter - matrix

matrix (ODMatrix *) - input

The new transform matrix for this transform.

SetMatrix Return Value - rv

rv (ODTransform *) - returns

A reference to this transform with its matrix replaced.

SetMatrix - Parameters

matrix (ODMatrix *) - input

The new transform matrix for this transform.

rv (ODTransform *) - returns

A reference to this transform with its matrix replaced.

SetMatrix - Remarks

If you specify an identity matrix, the effect of this method is the same as calling the [Reset](#) method.

If you use transforms of the [ODTransform](#) class and you have also created a subclass of [ODTransform](#) that applies complex transformation effects that cannot be represented by matrixes, your part needs to check whether a given transform has a matrix before calling this method. To do so, you should call the transform's [HasMatrix](#) method; only if that method returns kODTrue should you call the transform's SetMatrix method

SetMatrix - Exception Handling

kODErrTransformErr

The tranform has no inverse, or an overflow or underflow error occurred when calculating the inverse transform.

SetMatrix - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method must not call its inherited method; instead, it should raise an kODErrTransformErr exception. You can detect that a transform of your class has no matrix by calling its [HasMatrix](#) method.

SetMatrix - Related Methods

Related Methods

- [ODTransform::GetMatrix](#)
 - [ODTransform::HasMatrix](#)
 - [ODTransform::Reset](#)
-

SetMatrix - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

SetMATRIXLF (OS/2)

SetMATRIXLF (OS/2) - Syntax

This method sets the matrix to be identical to the specified GPI matrix.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
MATRIXLF      *mtx;
```

```
SetMATRIXLF (mtx);
```

SetMATRIXLF (OS/2) Parameter - mtx

mtx (MATRIXLF *) - input
The GPI matrix-elements structure.

SetMATRIXLF (OS/2) - Return Value

None.

SetMATRIXLF (OS/2) - Parameters

mtx (MATRIXLF *) - input
The GPI matrix-elements structure.

None.

SetMATRIXLF (OS/2) - Remarks

The translation component of the [MATRIXLF](#) structure is converted to a 16:16 fixed integer representation by shifting the values 16 bits to the left. No checking is done for overflow.

After this method executes successfully, you are responsible for deleting the [MATRIXLF](#) structure. This transform does not use or modify it.

SetMATRIXLF (OS/2) - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetOffset

SetOffset - Syntax

This method changes this transform into a pure offset with the specified horizontal and vertical translation values.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;
ODTransform  *rv;

rv = SetOffset(point);
```

SetOffset Parameter - point

point ([ODPoint *](#)) - input

A point specifying the new horizontal (x) and vertical (y) translation values.

SetOffset Return Value - rv

rv ([ODTransform *](#)) - returns

A reference to this transform changed to an offset transform.

SetOffset - Parameters

point ([ODPoint *](#)) - input
A point specifying the new horizontal (x) and vertical (y) translation values.

rv ([ODTransform *](#)) - returns
A reference to this transform changed to an offset transform.

SetOffset - Remarks

The translation values are the first two elements of the bottom row of a transform's matrix.

SetOffset - Related Methods

Related Methods

- [ODTransform::GetOffset](#)
-

SetOffset - Topics

Class:
[ODTransform](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

TransformPoint

TransformPoint - Syntax

This method should modify the location of the specified point by applying this transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;

TransformPoint(point);
```

TransformPoint Parameter - point

point ([ODPoint *](#)) - in/out

The point to be modified. On return, the fields of this structure have been modified to represent the transformed point.

TransformPoint - Return Value

None.

TransformPoint - Parameters

point ([ODPoint *](#)) - in/out

The point to be modified. On return, the fields of this structure have been modified to represent the transformed point.

None.

TransformPoint - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

TransformPoint - Related Methods

Related Methods

- [ODTransform::InvertPoint](#)

TransformPoint - Topics

Class:

ODTransform

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Override Policy](#)[Related Methods](#)

TransformPoints (OS/2)

TransformPoints (OS/2) - Syntax

This method transforms an array of points.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *points;
ODULong          npoints;

TransformPoints(points, npoints);
```

TransformPoints (OS/2) Parameter - points

points ([ODByteArray](#) *) - in/out

A pointer a byte array structure referencing an array of [ODPoint](#) structures.

TransformPoints (OS/2) Parameter - npoints

npoints ([ODULong](#)) - output

The number of points in the *points* .

TransformPoints (OS/2) - Return Value

None.

TransformPoints (OS/2) - Parameters

points ([ODByteArray *](#)) - in/out

A pointer a byte array structure referencing an array of [ODPoint](#) structures.

npoints ([ODULong](#)) - output

The number of points in the *points* .

None.

TransformPoints (OS/2) - Remarks

The points in the array are transformed in-place.

TransformPoints (OS/2) - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

TransformShape

TransformShape - Syntax

This method modifies the specified shape by applying this transform.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *shape;
```

```
TransformShape(shape);
```

TransformShape Parameter - shape

shape (ODShape *) - input
A reference to the shape object to be modified.

TransformShape - Return Value

None.

TransformShape - Parameters

shape (ODShape *) - input
A reference to the shape object to be modified.

None.

TransformShape - Remarks

Calling this method is equivalent to calling the [Transform](#) method of the specified shape.

TransformShape - Exception Handling

kODErrOutOfMemory

There is not enough memory to transform the shape.

TransformShape - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

TransformShape - Related Methods

Related Methods

- [ODShape::Transform](#)
- [ODTransform::InvertShape](#)

TransformShape - Topics

Class:

[ODTransform](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

WriteTo

WriteTo - Syntax

This method writes this transform's matrix to the specified storage unit.

```
#define INCL_ODTRANSFORM
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;

WriteTo(storageUnit);
```

WriteTo Parameter - storageUnit

storageUnit (ODStorageUnit *) - input

A reference to the storage unit where this transform's matrix is to be written.

WriteTo - Return Value

None.

WriteTo - Parameters

storageUnit (ODStorageUnit *) - input

A reference to the storage unit where this transform's matrix is to be written.

None.

WriteTo - Remarks

Before calling this method, you must focus the storage unit to the property where the matrix is to be written. The matrix writes the matrix into the value of type kODTransform in the focused property, replacing any matrix that was previously stored in that value or creating the value if it does not already exist.

WriteTo - Exception Handling

kODErrUnfocusedStorageUnit

This storage unit is not focused on a property or a value.

WriteTo - Override Policy

If you subclass [ODTransform](#) to create a nonlinear transform class, you must override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

WriteTo - Related Methods

Related Methods

- [ODTransform::ReadFrom](#)
-

WriteTo - Topics

Class:

ODTransform

Select an item:

ODTranslation

Class Definition File: TRANSLT.IDL

Class Hierarchy

SOMObject
 ODObject
 ODBaseTranslation
 ODTranslation

Description

An object of the ODTranslation class provides data translation services for OpenDoc documents and their parts.

The ODTranslation class depends on platform-specific system services to provide OpenDoc data translation. OpenDoc uses translation objects to maintain information on what kinds of translations are available to the user. OpenDoc and part editors can also use the translation object to perform any requested translations, rather than directly calling the underlying platform-specific services.

When a document is opened, the session object creates a single translation object. All parts of the document share the translation object; you can obtain a reference to it by calling the session object's [GetTranslation](#) method.

The translation service can be triggered when a part does not know how to handle data of an unfamiliar type (for example, when a user initiates linking or importing data from clipboard or drag-and-drop objects). The part can ask the translation object to translate the data into a recognizable format (part kind). Similarly, OpenDoc can initiate translation when opening a part on a system in which no part editor can directly read the part's data.

OpenDoc does not use platform types; therefore, to operate between OpenDoc and non-OpenDoc systems, platform types and ISO types need to be translated from one kind to the other. If you have part data expressed as a OS/2 file type and need to express it as a part kind (ISO string), you can use the translation object's [GetISOTypeFromPlatformType](#) method to find out if there is a part kind equivalent to that file type. To convert in the opposite direction, use the translation object's [GetPlatformTypeFromISOType](#) method, instead.

For more information related to data translation, see the chapter on data transfer in the *OpenDoc Programming Guide* .

Methods

The following list shows the methods defined by the ODTranslation class:

- [CanTranslate](#)
- [GetISOTypeFromPlatformType](#)
- [GetPlatformTypeFromISOType](#)
- [GetTranslationOf](#)
- [Translate](#)
- [TranslateView](#)

Overridden Methods

There are currently no methods overridden by the ODTranslation class.

CanTranslate

CanTranslate - Syntax

This method indicates whether translation is possible from the specified value type.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODValueType      fromType;
ODTranslateResult rv;

rv = CanTranslate(fromType);
```

CanTranslate Parameter - fromType

fromType (ODValueType) - input
The type of data to be translated.

CanTranslate Return Value - rv

rv (ODTranslateResult) - returns
The result of a translation. This parameter can return one of the following values:

kODCanTranslate
Translation is allowed with the specified type.
kODCannotTranslate
Translation is not allowed with the specified type.

CanTranslate - Parameters

fromType (ODValueType) - input
The type of data to be translated.

rv (ODTranslateResult) - returns
The result of a translation. This parameter can return one of the following values:

kODCanTranslate
Translation is allowed with the specified type.
kODCannotTranslate
Translation is not allowed with the specified type.

CanTranslate - Remarks

Your part calls this method to determine if translation facilities are available for a particular value type.

CanTranslate - Topics

Class:

ODTranslation

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

GetISOTypeFromPlatformType

GetISOTypeFromPlatformType - Syntax

This method returns the ISO type corresponding to the specified platform-specific type.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODPlatformType      platformType;
ODPlatformTypeSpace typeSpace;
ODValueType         rv;

rv = GetISOTypeFromPlatformType(platformType,
                                typeSpace);
```

GetISOTypeFromPlatformType Parameter - platformType

platformType ([ODPlatformType](#)) - input
A wrapper for the platform-specific type.

GetISOTypeFromPlatformType Parameter - typeSpace

typeSpace ([ODPlatformTypeSpace](#)) - input
The type of platform-specific structure identifying a type space (data or file). This parameter can be set to one of the following values:

- kODPlatformDataType
The native operating system scrap type.
- kODPlatformFileType
The native operating system file type.

GetISOTypeFromPlatformType Return Value - rv

rv ([ODValueType](#)) - returns
The corresponding ISO type.

GetISOTypeFromPlatformType - Parameters

platformType ([ODPlatformType](#)) - input
A wrapper for the platform-specific type.

typeSpace ([ODPlatformTypeSpace](#)) - input
The type of platform-specific structure identifying a type space (data or file). This parameter can be set to one of the following values:

[kODPlatformDataType](#)
The native operating system scrap type.
[kODPlatformFileType](#)
The native operating system file type.

rv ([ODValueType](#)) - returns
The corresponding ISO type.

GetISOTypeFromPlatformType - Remarks

Your part calls this method. OpenDoc does not use platform types; therefore, to interoperate between OpenDoc and non-OpenDoc systems, platform types and ISO types need to be translated from one kind to the other.

It is your responsibility to deallocate the returned value type when it is no longer needed.

GetISOTypeFromPlatformType - Related Methods

Related Methods

- [ODStorageSystem::CreatePlatformTypeList](#)
 - [ODTranslation::GetPlatformTypeFromISOType](#)
-

GetISOTypeFromPlatformType - Topics

Class:
[ODTranslation](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetPlatformTypeFromISOType

GetPlatformTypeFromISOType - Syntax

This method returns the platform-specific type corresponding to the specified ISO type.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODValueType      type;
ODPlatformType   rv;

rv = GetPlatformTypeFromISOType(type);
```

GetPlatformTypeFromISOType Parameter - type

type ([ODValueType](#)) - input
The ISO type.

GetPlatformTypeFromISOType Return Value - rv

rv ([ODPlatformType](#)) - returns
A wrapper for the corresponding platform-specific type.

GetPlatformTypeFromISOType - Parameters

type ([ODValueType](#)) - input
The ISO type.

rv ([ODPlatformType](#)) - returns
A wrapper for the corresponding platform-specific type.

GetPlatformTypeFromISOType - Remarks

Your part calls this method. OpenDoc does not use platform types; therefore, to interoperate between OpenDoc and non-OpenDoc systems, platform types and ISO types need to be translated from one kind to the other.

GetPlatformTypeFromISOType - Related Methods

Related Methods

- [ODStorageSystem::CreatePlatformTypeList](#)
 - [ODTranslation::GetISOTypeFromPlatformType](#)
-

GetPlatformTypeFromISOType - Topics

Class:

[ODTranslation](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

GetTranslationOf

GetTranslationOf - Syntax

This method returns a type list to which the specified value type can be translated.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODValueType      fromType;
ODTypeList       *rv;

rv = GetTranslationOf(fromType);
```

GetTranslationOf Parameter - fromType

fromType ([ODValueType](#)) - input

The type of data to be translated.

GetTranslationOf Return Value - rv

rv (ODTypeList *) - returns

A reference to a type list specifying a set of part kinds to which the specified value type can be translated or an empty list if the translation cannot be achieved.

GetTranslationOf - Parameters

fromType (ODValueType) - input

The type of data to be translated.

rv (ODTypeList *) - returns

A reference to a type list specifying a set of part kinds to which the specified value type can be translated or an empty list if the translation cannot be achieved.

GetTranslationOf - Remarks

Your part calls this method to determine all possible results that you can obtain by translating the specified type of data. This method does not change the value of the data to be translated.

GetTranslationOf - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the type list object.

This method may throw platform-specific exceptions.

GetTranslationOf - Topics

Class:

ODTranslation

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

Translate

Translate - Syntax

This method translates the specified source data to the specified value type.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODValueType      fromType;
ODByteArray      *fromData;
ODValueType      toType;
ODByteArray      *toData;
ODTranslateResult rv;

rv = Translate(fromType, fromData, toType,
               toData);
```

Translate Parameter - fromType

fromType (ODValueType) - input
The type of the source data to be translated.

Translate Parameter - fromData

fromData (ODByteArray *) - input
A byte array whose buffer contains the source data to be translated.

Translate Parameter - toType

toType (ODValueType) - input
The type to which the source data is to be translated.

Translate Parameter - toData

toData (ODByteArray *) - output
A byte array whose buffer is to contain the translated data.

Translate Return Value - rv

rv ([ODTranslateResult](#)) - returns

The result of a translation. This parameter can return one of the following values:

kODCanTranslate

Translation is allowed with the specified type.

kODCannotTranslate

Translation is not allowed with the specified type.

Translate - Parameters

fromType ([ODValueType](#)) - input

The type of the source data to be translated.

fromData ([ODByteArray *](#)) - input

A byte array whose buffer contains the source data to be translated.

toType ([ODValueType](#)) - input

The type to which the source data is to be translated.

toData ([ODByteArray *](#)) - output

A byte array whose buffer is to contain the translated data.

rv ([ODTranslateResult](#)) - returns

The result of a translation. This parameter can return one of the following values:

kODCanTranslate

Translation is allowed with the specified type.

kODCannotTranslate

Translation is not allowed with the specified type.

Translate - Remarks

Your part calls this method after calling the [CanTranslate](#) method to establish that the source type can be translated. This method does not change the content of the source byte array.

If translation is successful, this method allocates the destination byte array structure and its buffer, and stores the translated data in that buffer. It is your responsibility to deallocate the byte array structure (and its buffer) when it is no longer needed.

Translate - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the destination byte array structure (or its buffer).

This method may throw platform-specific exceptions.

Translate - Related Methods

Related Methods

- [ODTranslation::CanTranslate](#)

Translate - Topics

Class:

ODTranslation

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

TranslateView

TranslateView - Syntax

This method translates the content of the source storage-unit view and stores the translated data in the destination storage-unit view.

```
#define INCL_ODTRANSLATION
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *fromView;
ODStorageUnitView    *toView;
ODTranslateResult     rv;

rv = TranslateView(fromView, toView);
```

TranslateView Parameter - fromView

fromView (ODStorageUnitView *) - input

A reference to a storage-unit view; the focused value that contains the translated data.

TranslateView Parameter - toView

toView (ODStorageUnitView *) - input

A reference to a storage-unit view; the focused value that is to contain the translated data.

TranslateView Return Value - rv

rv ([ODTranslateResult](#)) - returns

The result of a translation. This parameter can return one of the following values:

kODCanTranslate

Translation is allowed with the specified type.

kODCannotTranslate

Translation is not allowed with the specified type.

TranslateView - Parameters

fromView (ODStorageUnitView *) - input

A reference to a storage-unit view; the focused value that contains the translated data.

toView (ODStorageUnitView *) - input

A reference to a storage-unit view; the focused value that is to contain the translated data.

rv ([ODTranslateResult](#)) - returns

The result of a translation. This parameter can return one of the following values:

kODCanTranslate

Translation is allowed with the specified type.

kODCannotTranslate

Translation is not allowed with the specified type.

TranslateView - Remarks

Your part calls this method when it wants to translate some data in a storage unit.

TranslateView - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the translated data.

This method may throw platform-specific exceptions.

TranslateView - Topics

Class:

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

ODTypeList

Class Definition File: TYPELIST.IDL

Class Hierarchy

SOMObject
 ODObject
 ODTypeList

Description

An object of the ODTypeList class is an ordered set of [ODType](#) elements.

A type list is an ordered set of elements, each specifying a different value. Elements are of the [ODType](#) type and so are strings that can specify part kinds, focus types, or storage-unit types. Most often, all elements of a type list are part kinds. Because an [ODType](#) value is a pointer to an ISO string, any value to be added to a type list is copied first. That is, the pointer itself is not added to the list; instead, the string is copied and a pointer to the new copy is added to the list.

To create a type list, call the [CreateTypeList](#) method of the storage-system object. If the call to this method specifies an existing type list, the new type list is initialized to contain a copy of each elements in that list. The elements in the new type list are in the same order as the element of the original list. Otherwise, the new list is initialized to an empty list (a list with no elements).

You can add elements, one at a time, to the end of the type list. OpenDoc ensures that each element of a type list is unique; if you attempt to add a value that is already in the list, the list remains unchanged. You can remove elements from the list, test whether the list contains a particular element, and get the number of elements in the list. If you need to perform an operation for each element of the list, you can create an object of the [ODTypeListIterator](#) class and use it to iterate through the list.

Methods

The following list shows the methods defined by the ODTypeList class:

- [AddLast](#)
- [Contains](#)
- [Count](#)
- [CreateTypeListIterator](#)
- [Remove](#)

Overridden Methods

There are currently no methods overridden by the ODTypeList class.

AddLast

AddLast - Syntax

This method adds an element to the end of this type list.

```
#define INCL_ODTYPELIST
```

```
#define INCL_ODAPI
#include <os2.h>

ODType    type;

AddLast (type);
```

AddLast Parameter - type

type (ODType) - input
The element to be added to the list.

AddLast - Return Value

None.

AddLast - Parameters

type (ODType) - input
The element to be added to the list.

None.

AddLast - Remarks

If this type list already contains an element equal to the specified element, no action is taken; otherwise, a copy of the specified element is added to the end of the list.

AddLast - Exception Handling

kODErrOutOfMemory

There is not enough memory to add the specified element to this type list.

AddLast - Related Methods

Related Methods

- [ODTypeList::Contains](#)
- [ODTypeList::Remove](#)

AddLast - Topics

Class:

ODTypeList

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

Contains

Contains - Syntax

This method indicates whether this type list contains a specified element.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODType      type;
ODBoolean   rv;
```

```
rv = Contains(type);
```

Contains Parameter - type

type ([ODType](#)) - input

The element to be tested for inclusion in this list.

Contains Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the this type list contains an element equal to the specified element.

kODTrue	This type list contains an element equal to the specified element.
kODFalse	This type list does not contain an element equal to the specified element.

Contains - Parameters

type ([ODType](#)) - input
The element to be tested for inclusion in this list.

rv ([ODBoolean](#)) - returns
A flag indicating whether the this type list contains an element equal to the specified element.

kODTrue	This type list contains an element equal to the specified element.
kODFalse	This type list does not contain an element equal to the specified element.

Contains - Related Methods

Related Methods

- [ODTypeList::AddLast](#)
- [ODTypeList::Remove](#)

Contains - Topics

Class:
[ODTypeList](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Related Methods](#)

Count

Count - Syntax

This method returns the number of elements in this type list.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
#include <os2.h>

ODULong    rv;

rv = Count();
```

Count Return Value - rv

rv ([ODULong](#)) - returns
The number of elements in this type list or 0 if the list is empty.

Count - Parameters

rv ([ODULong](#)) - returns
The number of elements in this type list or 0 if the list is empty.

Count - Topics

Class:
ODTypeList

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

CreateTypeListIterator

CreateTypeListIterator - Syntax

This method creates a type-list iterator for this type list.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
```

```
#include <os2.h>

ODTypeListIterator      *rc;

rc = CreateTypeListIterator();
```

CreateTypeListIterator Return Value - rc

rc (ODTypeListIterator *) - returns
A reference to the newly created type-list iterator.

CreateTypeListIterator - Parameters

rc (ODTypeListIterator *) - returns
A reference to the newly created type-list iterator.

CreateTypeListIterator - Remarks

You call this method if you need to apply an operation to each element of this type list.

While you are using the returned type-list iterator, you must not modify this type list; in particular, you must not add or remove elements, and you must not delete this type list.

You must delete the returned type-list iterator when it is no longer needed.

CreateTypeListIterator - Exception Handling

kODErrOutOfMemory	There is not enough memory to create the iterator.
-------------------	--

CreateTypeListIterator - Topics

Class:
ODTypeList

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

Remove

Remove - Syntax

This method removes a specified element from this type list.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODType    type;
```

```
Remove (type);
```

Remove Parameter - type

type (ODType) - input
The element to be removed.

Remove - Return Value

None.

Remove - Parameters

type (ODType) - input
The element to be removed.

None.

Remove - Remarks

If this type list contains an element equal to the specified type, that element is removed from the list; if not, no action is taken.

Remove - Related Methods

Related Methods

- [ODTypeList::AddLast](#)
 - [ODTypeList::Contains](#)
-

Remove - Topics

Class:

[ODTypeList](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

ODTypeListIterator

Class Definition File: TYPLSITR.IDL

Class Hierarchy

SOMObject
 ODObject
 ODTypeListIterator

Description

An object of the ODTypeListIterator class provides access to each element of a type list.

You use a type-list iterator to apply an operation to each element of a type list. For example, a part might use a type-list iterator to enumerate the part kinds that the part supports and write to storage a representation for each one.

Methods of the ODTypeListIterator class return copies of the elements in the type list. This design prevents you from accidentally changing the content of the type list. When you use a type-list iterator, be sure to delete each copied string to avoid a memory leak.

Your part creates a type-list iterator object by calling the type list object's [CreateTypeListIterator](#) method, which returns a reference to a type-list iterator object.

While you are using a type-list iterator, you should not modify or delete the type list that created it. You must postpone adding elements to or removing elements from the type list until after you have deleted the iterator.

For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

Methods

The methods defined by the ODTypeListIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

Overridden Methods

There are currently no methods overridden by the ODTypeListIterator class.

First

First - Syntax

This method begins the iteration and returns a copy of the first element in the type list that created this type-list iterator.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODType    rv;
```

```
rv = First();
```

First Return Value - rv

rv (ODType) - returns

A copy of the first element in the type list or KODNULL if the type list is empty.

First - Parameters

rv (ODType) - returns

A copy of the first element in the type list or KODNULL if the type list is empty.

First - Remarks

Your part must call this method before calling this type-list iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time resets the iteration.

It is your responsibility to deallocate the returned type value when it is no longer needed.

First - Exception Handling

kODErrIteratorOutOfSync

The type list was modified while the iteration was in progress.

kODErrOutOfMemory

There is not enough memory to create the type.

First - Topics

Class:

ODTypeListIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

IsNotComplete

IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

IsNotComplete Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

IsNotComplete - Parameters

rv ([ODBoolean](#)) - returns
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

IsNotComplete - Remarks

Your part calls this method to test whether more elements remain in the type list. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found an element. This method returns kODFalse when you have examined all the elements (that is, when the previous call to [First](#) or [Next](#) returned kODNULL). If the type list that created this iterator is empty, this method always returns kODFalse.

IsNotComplete - Exception Handling

kODErrIteratorNotInitialized	This method was called before calling either the First or Next method to begin the iteration.
kODErrIteratorOutOfSync	The type list was modified while the iteration was in progress.

IsNotComplete - Topics

Class:
[ODTypeListIterator](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

Next

Next - Syntax

This method returns a copy of the next element in the type list that created this type-list iterator.

```
#define INCL_ODTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODType rv;  
  
rv = Next();
```

Next Return Value - rv

rv ([ODType](#)) - returns
A copy of the next element in the type list or kODNULL if you have reached the end of the type list.

Next - Parameters

rv ([ODType](#)) - returns
A copy of the next element in the type list or kODNULL if you have reached the end of the type list.

Next - Remarks

If your part calls this method before calling this type-list iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

It is your responsibility to deallocate the returned type value when it is no longer needed.

Next - Exception Handling

kODErrIteratorOutOfSync	The type list was modified while the iteration was in progress.
kODErrOutOfMemory	There is not enough memory to create the type.

Next - Topics

Class:
[ODTypeListIterator](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

ODUndo

Class Definition File: UNDO.IDL

Class Hierarchy

SOMObject
 ODOObject
 ODUndo

Description

An object of the ODUndo class holds command history information to support the undo capability-the ability to reverse the effects of recently executed commands-of OpenDoc.

Your part editor stores undoable actions in, and retrieves them from, the undo object. When a document is opened, the session object creates a single undo object. All parts of that document share the undo object; you can obtain a reference to it by calling the session object's [GetUndo](#) method.

The undo object contains an undo stack and a redo stack. When an undoable action is performed, the part involved places *action data* on the undo stack. Action data is information provided by the part that allows it to reverse the effects of an undoable action. OpenDoc stores the action data in the undo object's *action history* -the cumulative set of reversible actions available at any one time. When an action needs to be undone, OpenDoc pops the action data from the undo stack onto the redo stack. At the same time, OpenDoc notifies your part so that your part can undo the recently executed action using the stored action data. When an undone action needs to be redone, OpenDoc pops the action data from the redo stack back onto the undo stack. OpenDoc notifies your part so that your part can redo the recently undone action using the stored action data. The undo and redo stacks can be cleared upon a part's request. When clearing is needed, OpenDoc asks each part to dispose of its action data stored in the stacks. The order of disposal is from newer actions to older actions.

There are times when an action subhistory is useful. An *action subhistory* a subset of reversible actions available at any one time. For example, you may need a new action context when entering a modal state, that is, when your part displays a modal dialog box. OpenDoc allows marks to be placed on the stacks. If a mark is placed on a stack, the stack may be cleared only to the mark. For example, when the modal dialog box closes, any actions done within the context of the modal dialog box are disposed of; however, all the actions executed before the modal dialog box appeared are preserved in the stacks.

Methods

The methods defined by the ODUndo class include:

- [AbortCurrentTransaction](#)
- [AddActionToHistory](#)
- [ClearActionHistory](#)
- [ClearRedoHistory](#)
- [MarkActionHistory](#)
- [PeekRedoHistory](#)
- [PeekUndoHistory](#)
- [Redo](#)
- [Undo](#)

Overridden Methods

There are currently no methods overridden by the ODUndo class.

AbortCurrentTransaction

AbortCurrentTransaction - Syntax

This method removes the current transaction (any nested transactions) from the action history.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
AbortCurrentTransaction();
```

AbortCurrentTransaction - Return Value

None.

AbortCurrentTransaction - Parameters

None.

AbortCurrentTransaction - Remarks

This method aborts a transaction that is being placed in the undo stack by removing all single actions up to and including the last begin action. If there is a nested transaction in the current transaction, it is entirely removed. This method in turn calls your part's [UndoAction](#) method to give your part the opportunity to perform any reverse editing necessary to restore itself to the state it possessed before the transaction began.

AbortCurrentTransaction - Related Methods

Related Methods

- [ODPart::UndoAction](#)

AbortCurrentTransaction - Topics

Class:

ODUndo

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

AddActionToHistory

AddActionToHistory - Syntax

This method pushes the action data and its associated part onto the undo stack.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>

ODPart      *whichPart;
ODActionData *actionData;
ODActionType actionType;
ODName      *undoActionLabel;
ODName      *redoActionLabel;

AddActionToHistory(whichPart, actionData,
                  actionType, undoActionLabel, redoActionLabel);
```

AddActionToHistory Parameter - whichPart

whichPart (ODPart *) - input

A reference to the part that performed the action.

AddActionToHistory Parameter - actionData

actionData (ODActionData *) - input

A byte array whose buffer contains the data needed by the part to allow it to undo the action.

AddActionToHistory Parameter - actionType

actionType (ODActionType) - input

The type of undo action. This parameter can be set to one of the following values:

kODBeginAction	The first action of a multistep action.
kODEndAction	The last action of a multistep action.
kODSingleAction	A single action.

AddActionToHistory Parameter - undoActionLabel

undoActionLabel (ODName *) - input
A user-visible label for the undo command beginning with the word Undo.

AddActionToHistory Parameter - redoActionLabel

redoActionLabel (ODName *) - input
A user-visible label for the redo command beginning with the word Redo.

AddActionToHistory - Return Value

None.

AddActionToHistory - Parameters

whichPart (ODPart *) - input
A reference to the part that performed the action.

actionData (ODActionData *) - input
A byte array whose buffer contains the data needed by the part to allow it to undo the action.

actionType (ODActionType) - input
The type of undo action. This parameter can be set to one of the following values:

- kODBBeginAction The first action of a multistep action.
- kODEndAction The last action of a multistep action.
- kODSingleAction A single action.

undoActionLabel (ODName *) - input
A user-visible label for the undo command beginning with the word Undo.

redoActionLabel (ODName *) - input
A user-visible label for the redo command beginning with the word Redo.

None.

AddActionToHistory - Exception Handling

kODErrCannotAddAction	The specified action cannot be added to this undo object; an undo or redo action is already in progress.
-----------------------	--

AddActionToHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

ClearActionHistory

ClearActionHistory - Syntax

This method clears the undo and redo stacks.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
ODRespectMarksChoices    respectMarks;
```

```
ClearActionHistory(respectMarks);
```

ClearActionHistory Parameter - respectMarks

respectMarks ([ODRespectMarksChoices](#)) - input

The values for clearing an action history. This parameter can be set to one of the following values:

kODDontRespectMarks

The stacks are cleared in their entirety.

kODRespectMarks

The stacks are cleared only down to the specified marks-that is, only actions within an action subhistory are cleared.

ClearActionHistory - Return Value

None.

ClearActionHistory - Parameters

respectMarks ([ODRespectMarksChoices](#)) - input

The values for clearing an action history. This parameter can be set to one of the following values:

kODDontRespectMarks

The stacks are cleared in their entirety.

kODRespectMarks

The stacks are cleared only down to the specified marks-that is, only actions within an action subhistory are cleared.

None.

ClearActionHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ClearRedoHistory

ClearRedoHistory - Syntax

This method clears the redo history.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
ClearRedoHistory();
```

ClearRedoHistory - Return Value

None.

ClearRedoHistory - Parameters

None.

ClearRedoHistory - Remarks

OpenDoc calls this method. If the redo stack contains a mark indicating an action subhistory, this method clears only that subhistory. Otherwise, it clears the entire redo stack.

ClearRedoHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

MarkActionHistory

MarkActionHistory - Syntax

This method marks the top of the undo and redo stacks.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
MarkActionHistory();
```

MarkActionHistory - Return Value

None.

MarkActionHistory - Parameters

None.

MarkActionHistory - Remarks

The marks are used to indicate the beginning of a new action subhistory in each stack.

MarkActionHistory - Exception Handling

kODErrCannotMarkAction

Failure to start an action subhistory by placing a mark at the beginning of the undo and redo stacks; the undo object was initialized properly.

MarkActionHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

PeekRedoHistory

PeekRedoHistory - Syntax

This method indicates whether there is anything on the redo stack and returns the information about the action at the top of the redo stack.

```

#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>

ODPart          **part;
ODActionData    *actionData;
ODActionType    *actionType;
ODName          *actionLabel;
ODBoolean       rv;

rv = PeekRedoHistory(part, actionData, actionType,
                    actionLabel);

```

PeekRedoHistory Parameter - part

part (ODPart **) - output
 A reference to the part that performed the action at the top of the redo stack.

PeekRedoHistory Parameter - actionData

actionData (ODActionData *) - output
 A byte array whose buffer is to contain the action data for the action at the top of the redo stack.

PeekRedoHistory Parameter - actionType

actionType (ODActionType *) - output
 The type of undo action. This parameter can be set to one of the following values:

- kODBBeginAction The first action of a multistep action.
- kODEndAction The last action of a multistep action.
- kODSingleAction A single action.

PeekRedoHistory Parameter - actionLabel

actionLabel (ODName *) - output
 A user-visible label for the redo command beginning with the word Redo.

PeekRedoHistory Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether there is anything on the redo stack.

kODTrue

There is something on the redo stack.

kODFalse

The redo stack is empty or the top of the redo stack contains a mark indicating an action subhistory.

PeekRedoHistory - Parameters

part ([ODPart **](#)) - output

A reference to the part that performed the action at the top of the redo stack.

actionData ([ODActionData *](#)) - output

A byte array whose buffer is to contain the action data for the action at the top of the redo stack.

actionType ([ODActionType *](#)) - output

The type of undo action. This parameter can be set to one of the following values:

kODBeginAction

The first action of a multistep action.

kODEndAction

The last action of a multistep action.

kODSingleAction

A single action.

actionLabel ([ODName *](#)) - output

A user-visible label for the redo command beginning with the word Redo.

rv ([ODBoolean](#)) - returns

A flag indicating whether there is anything on the redo stack.

kODTrue

There is something on the redo stack.

kODFalse

The redo stack is empty or the top of the redo stack contains a mark indicating an action subhistory.

PeekRedoHistory - Remarks

The document shell or container applications call this method to properly set up the redo item. Your part can also call this method, but it is typically unnecessary.

PeekRedoHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

PeekUndoHistory

PeekUndoHistory - Syntax

This method indicates whether there is anything on the undo stack and returns information about the action at the top of the undo stack.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>

ODPart          **part;
ODActionData    *actionData;
ODActionType     *actionType;
ODName          *actionLabel;
ODBoolean        rv;

rv = PeekUndoHistory(part, actionData, actionType,
                    actionLabel);
```

PeekUndoHistory Parameter - part

part (ODPart **) - output
A reference to the part that performed the action at the top of the undo stack.

PeekUndoHistory Parameter - actionData

actionData (ODActionData *) - output
A byte array whose buffer is to contain the action data for the action at the top of the undo stack.

PeekUndoHistory Parameter - actionType

actionType (ODActionType *) - output
The values for an undo action. This parameter can be set to one of the following values:

kODBeginAction	The first action of a multistep action.
kODEndAction	The last action of a multistep action.
kODSingleAction	A single action.

PeekUndoHistory Parameter - actionLabel

actionLabel (ODName *) - output
The user-visible label for the undo command beginning with the word Undo.

PeekUndoHistory Return Value - rv

rv (ODBoolean) - returns
A flag indicating whether there is anything on the undo stack.

kODTrue	There is something on the undo stack.
kODFalse	The undo stack is empty or the top of the undo stack contains a mark indicating an action subhistory.

PeekUndoHistory - Parameters

part (ODPart **) - output
A reference to the part that performed the action at the top of the undo stack.

actionData (ODActionData *) - output
A byte array whose buffer is to contain the action data for the action at the top of the undo stack.

actionType (ODActionType *) - output
The values for an undo action. This parameter can be set to one of the following values:

kODBeginAction	The first action of a multistep action.
kODEndAction	Te last action of a multistep action.
kODSingleAction	A single action.

actionLabel (ODName *) - output
The user-visible label for the undo command beginning with the word Undo.

rv (ODBoolean) - returns
A flag indicating whether there is anything on the undo stack.

kODTrue	There is something on the undo stack.
kODFalse	The undo stack is empty or the top of the undo stack contains a mark indicating an action subhistory.

PeekUndoHistory - Remarks

The document shell or container applications call this method to properly set up the undo item. Your part can also call this method, but it is typically unnecessary.

PeekUndoHistory - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

Redo

Redo - Syntax

This method is called by the document shell or container application to redo the top action in the redo history.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
Redo();
```

Redo - Return Value

None.

Redo - Parameters

None.

Redo - Exception Handling

kODErrEmptyStack

The redo stack is empty or the undo object was not initialized properly.

This method may throw OSA event exceptions or any other exceptions returned by the part.

Redo - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

Undo

Undo - Syntax

This method is called by the document shell or container application to undo the top action in the undo stack.

```
#define INCL_ODUNDO
#define INCL_ODAPI
#include <os2.h>
```

```
Undo();
```

Undo - Return Value

None.

Undo - Parameters

None.

Undo - Exception Handling

kODErrEmptyStack

The undo stack is empty or the undo object was not initialized properly.

This method may throw OSA event exceptions or any other exceptions returned by the part.

Undo - Topics

Class:

ODUndo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

ODValueIterator

Class Definition File: VALUEITR.IDL

Class Hierarchy

SOMObject

ODObject

ODValueIterator

Description

An object of the ODValueIterator class provides access to the entries of a value name space.

You use a value iterator to apply an operation to all entries of a value name space. For example, you might use a value iterator to iterate over a list of part editors to obtain information about each part editor, such as the part's supported part kinds.

Your part creates a value iterator object by calling the value name space's [CreateIterator](#) method, which returns a reference to a value iterator object. The iterator performs an unordered traversal of the name space.

While you are using a value iterator, you should not modify or delete the name space. You must postpone adding entries to or removing entries from the name space until after you have deleted the iterator.

For more information related to value name spaces, see the class description for [ODValueNameSpace](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

Methods

The methods defined by the ODValueIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

Overridden Methods

There are currently no methods overridden by the ODValueIterator class.

First

First - Syntax

This method begins the iteration and returns the first entry in the name space.

```
#define INCL_ODVALUEITERATOR
#define INCL_ODAPI
#include <os2.h>

ODISOStr      *key;
ODByteArray   *value;

First(key, value);
```

First Parameter - key

key (ODISOStr *) - in/out

A pointer to an ISO string representing the key in the first entry in the name space or KODNULL for an empty name space.

First Parameter - value

value (ODByteArray *) - in/out

A byte array whose buffer contains the value for the first entry in the name space.

First - Return Value

None.

First - Parameters

key (ODISOStr *) - in/out

A pointer to an ISO string representing the key in the first entry in the name space or KODNULL for an empty name space.

value (ODByteArray *) - in/out

A byte array whose buffer contains the value for the first entry in the name space.

None.

First - Remarks

Your part must call this method before calling this value iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

It is your responsibility to deallocate the ISO string and byte array structure (and its buffer) when they are no longer needed. You must also delete the key when it is no longer needed.

First - Exception Handling

kODErrIteratorOutOfSync

The name space was modified while the iteration was in progress.

First - Topics

Class:

ODValueIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

IsNotComplete

IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODVALUEITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

IsNotComplete Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

IsNotComplete - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

IsNotComplete - Remarks

Your part calls this method to test whether more entries remain in the name space. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found an entry. This method returns kODFalse when you have examined all the entries (that is, when the previous call to [First](#) or [Next](#) returned kODNULL). If the name space is empty, this method always returns kODFalse.

IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The name space was modified while the iteration was in progress.

IsNotComplete - Topics

Class:

ODValueIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

Next

Next - Syntax

This method returns the next entry in the name space.

```
#define INCL_ODVALUEITERATOR
#define INCL_ODAPI
#include <os2.h>

ODISOStr      *key;
ODByteArray   *value;

Next(key, value);
```

Next Parameter - key

key (ODISOStr *) - in/out

A pointer to an ISO string representing the key in the next entry in the name space or KODNULL if you have reached the end of the name space.

Next Parameter - value

value (ODByteArray *) - in/out

A byte array whose buffer contains the value for the next entry in the name space.

Next - Return Value

none.

Next - Parameters

key ([ODISOStr *](#)) - in/out

A pointer to an ISO string representing the key in the next entry in the name space or kODNULL if you have reached the end of the name space.

value ([ODByteArray *](#)) - in/out

A byte array whose buffer contains the value for the next entry in the name space.

none.

Next - Remarks

If your part calls this method before calling this value iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

It is your responsibility to deallocate the ISO string and byte array structure (and its buffer) when they are no longer needed. You must also delete the key when it is no longer needed.

Next - Exception Handling

kODErrIteratorOutOfSync

The name space was modified while the iteration was in progress.

Next - Topics

Class:

ODValueIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

ODValueNameSpace

Class Definition File: VALUENS.IDL

Class Hierarchy

SOMObject

SOMClass

 ODOObject

 ODNameSpace

ODValueNameSpace

Description

An object of the ODValueNameSpace class is a collection of values, each of which has a unique key to identify the value within the collection.

A value name space allows a part to identify a value using a unique key, which can be passed easily between parts. A part can create a value name space to store information about other part editors, such as the part's supported part kinds. The values in a value name space are

stored as byte arrays whose buffers can contain values of any types.

Your part can create objects of the ODValueNameSpace class by calling the name-space manager's [CreateNameSpace](#) method, passing the constant kODNSDataTypeODValue for the type of the name space.

Value name spaces can be arranged hierarchically to allow you to search multiple value name spaces for a single key. Searches move from a child value name space to its parent value name space until the entry is found or until there are no more value name spaces to search.

Methods

The methods defined by the ODValueNameSpace class include:

- [CreateIterator](#)
- [GetEntry](#)
- [Register](#)

Overridden Methods

There are currently no methods overridden by the ODValueNameSpace class.

CreateIterator

CreateIterator - Syntax

This method creates a value iterator for the entries in this value name space.

```
#define INCL_ODVALUENAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODValueIterator    *rv;

rv = CreateIterator();
```

CreateIterator Return Value - rv

rv (ODValueIterator *) - returns
A reference to a value iterator for iterating over the entries in this value name space.

CreateIterator - Parameters

rv (ODValueIterator *) - returns
A reference to a value iterator for iterating over the entries in this value name space.

CreateIterator - Remarks

You call this method when you need to apply an operation to all the entries of this name space.

While you are using the returned value iterator, you must not modify this value name space; in particular, you must not register or unregister entries, and you must not delete this value name space.

You must delete the iterator when it is no longer needed.

CreateIterator - Topics

Class:

ODValueNameSpace

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

GetEntry

GetEntry - Syntax

This method searches for an entry with the specified key and, if it exists, returns the value associated with that key.

```
#define INCL_ODVALUENAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODISOStr      key;
ODByteArray    *value;
ODBoolean      rv;
```

```
rv = GetEntry(key, value);
```

GetEntry Parameter - key

key ([ODISOStr](#)) - input

The key to search for in this value name space.

GetEntry Parameter - value

value ([ODByteArray *](#)) - output

The byte array whose buffer is to contains the value corresponding to the specified key, if the entry is found.

GetEntry Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether the entry was found.

TRUE

The entry is found.

FALSE

The entry is not found.

GetEntry - Parameters

key ([ODISOStr](#)) - input

The key to search for in this value name space.

value ([ODByteArray *](#)) - output

The byte array whose buffer is to contains the value corresponding to the specified key, if the entry is found.

rv ([ODBoolean](#)) - returns

A flag indicating whether the entry was found.

TRUE

The entry is found.

FALSE

The entry is not found.

GetEntry - Remarks

If the specified key is found, this method copies the entry's associated value into the *value* parameter. If the specified key is not found in this name space, this method searches the parent name space. Searches proceed from each value name space to its parent until one of the following happens: the entry is found, there is no parent name space to search, or the parent name space is an object name space instead of a value name space.

If the key is found, the *_buffer* field of *value* output parameter is to point to a memory block containing the copy of the value associated with the key. If the key is not found after searching this value name space and its ancestors, the *_buffer* field of *value* parameter is set to KODNULL.

Your part must delete the byte array and its buffer when they are no longer needed.

GetEntry - Related Methods

Related Methods

- [ODNameSpace::Exists](#)
- [ODValueNameSpace::Register](#)

GetEntry - Topics

Class:
ODValueNameSpace

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

Register

Register - Syntax

This method adds a new entry to this value name space.

```
#define INCL_ODVALUENAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODISOStr      key;
ODByteArray   *value;

Register(key, value);
```

Register Parameter - key

key ([ODISOStr](#)) - input
The key for the new entry.

Register Parameter - value

value ([ODByteArray *](#)) - input
The byte array whose buffer contains the value to associate with the key.

Register - Return Value

None.

Register - Parameters

key ([ODISOSTr](#)) - input
The key for the new entry.

value ([ODByteArray *](#)) - input
The byte array whose buffer contains the value to associate with the key.

None.

Register - Remarks

This method copies both the *key* and *value* paramter's buffer into a new entry in the name space. If the specified key already exists within this name space, this method overwrites the old value with the new one and deallocates the memory used by the old value.

Register - Related Methods

Related Methods

- [ODNameSpace::Unregister](#)
-

Register - Topics

Class:
[ODValueNameSpace](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

ODViewExtension

Class Definition File: .IDL

Class Hierarchy

SOMObject
 ODObject
 ODRefCntObject
 ODEExtension
 ODViewExtension

Description

All part handlers that are containers should support tree, icon, and details views. These views can be invoked by the user on the active part from the **View** menu. The ODViewExtension class provides part developers with an interface that supports these views.

The ODViewExtension class is accessible by part handlers. The default implementation of the ODViewExtension class allows a containing part to display tree, icon, and details views of its embedded part's hierarchy similar to the way the Workplace Shell displays directories and files in its implementation of views.

Tree view	Displays a tree hierarchy of parts. The part upon which the user selected tree view appears as the root of the tree. Initially, the tree appears with only the first branch of embedded parts showing.
Icon view	Displays embedded parts as icons.
Details view	Displays a list of embedded parts along with their properties information. The part handler upon which the view was selected can add additional columns and data to this view by subclassing.

At the simplest level, a part developer needs only to create an instance of the ODViewExtension object, initialize it, and invoke the [DisplayView](#) method passing the view type that the user selected. OpenDoc creates the specified view window by iterating through the embedded parts to obtain its part information, displaying the view, and restoring and saving user-specified attributes, such as window size and position, color, and font. When a part no longer needs the ODViewExtension object, it should delete it.

At a more complex level, a part developer can subclass the ODViewExtension class and override the display method to create a custom view for the part. The part developer is then responsible for all the underlying code that is necessary to display and support embedded parts in this view, handle all PM related notifications, and route the notifications to the appropriate part, if necessary. A part overriding the [DisplayView](#) method should still forward the default views to its parent for handling.

Some container parts may have additional information to display for their embedded parts. A part can add these additional columns into its details view by calling the [AddDetailsColumns](#) method, passing in a filled-in structure containing the storage unit to be read, the data type to be displayed, and the column heading. OpenDoc then sets up the additional columns for viewing and retrieving the data from the part's storage unit. The additional columns can contain data, times, strings, numerics or icons. Bit maps, while allowed by the container control for column headings and data, cannot be passed to an ODViewExtension object to be included in details view. The PM container control does not support both bit maps and icons within the same view, and OpenDoc is using icons for the first column display type. A part developer is free to subclass ODViewExtension to display bit maps in a details view.

Three standard view pages are added to the Part Properties notebook, one page for each of the specified views. Users are able to modify the attributes of each view. These notebook pages are identical to those pages used by the Workplace Shell for view customization. If a part developer creates his own custom view, a notebook page should be added for use customization, if applicable, and persistently store the user's preferences.

Methods

The methods defined by the ODViewExtension class include:

- [AddDetailsColumns](#)
- [DisplayView](#)
- [InitViewExtension](#)

Overridden Methods

There are currently no methods overridden by the ODViewExtension class.

AddDetailsColumns (OS/2)

AddDetailsColumns (OS/2) - Syntax

This method displays additional information in details view.

```

#define INCL_ODVIEWEXTENSION
#define INCL_ODAPI
#include <os2.h>

ODFIELDINFO      *pPartODFieldInfo;
char              **sColDataType;
char              **sPartSUPropName;
char              **sPartSUValName;
ODULong           ulNumberOfColumns;
ODBoolean         rv;

rv = AddDetailsColumns(pPartODFieldInfo, sColDataType,
                      sPartSUPropName, sPartSUValName, ulNumberOfColumns);

```

AddDetailsColumns (OS/2) Parameter - pPartODFieldInfo

pPartODFieldInfo (ODFIELDINFO *) - input

An array of field information structures containing additional information to be added about data and columns. Flags for items in this structure can be any of the CFA_* values defined for PM container control windows.

AddDetailsColumns (OS/2) Parameter - sColDataType

sColDataType (char **) - input

An array of strings containing an OpenDoc data type for storage unit data retrieval. Each item in the array corresponds to an element in the *pPartODFieldInfo* parameter. The data types must be OpenDoc data types, equivalent to the CFA_* data types specified in the ODFIELDINFO structure.

AddDetailsColumns (OS/2) Parameter - sPartSUPropName

sPartSUPropName (char **) - input

An array of strings containing the part's storage unit property name where the data can be found.

AddDetailsColumns (OS/2) Parameter - sPartSUValName

sPartSUValName (char **) - input

An array of strings containing the part's storage unit value name where the data can be found.

AddDetailsColumns (OS/2) Parameter - ulNumberOfColumns

ulNumberOfColumns ([ODULong](#)) - input

The number of additional columns being requested. This number is also the number of elements passed in by each of the three arrays above.

AddDetailsColumns (OS/2) Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indication the success.

kODTrue	Successful completion
kODFalse	Error occurred

AddDetailsColumns (OS/2) - Parameters

pPartODFieldInfo ([ODFIELDINFO *](#)) - input

An array of field information structures containing additional information to be added about data and columns. Flags for items in this structure can be any of the CFA_* values defined for PM container control windows.

sColDataType (char **) - input

An array of strings containing an OpenDoc data type for storage unit data retrieval. Each item in the array corresponds to an element in the *pPartODFieldInfo* parameter. The data types must be OpenDoc data types, equivalent to the CFA_* data types specified in the [ODFIELDINFO](#) structure.

sPartSUPropName (char **) - input

An array of strings containing the part's storage unit property name where the data can be found.

sPartSUValName (char **) - input

An array of strings containing the part's storage unit value name where the data can be found.

ulNumberOfColumns ([ODULong](#)) - input

The number of additional columns being requested. This number is also the number of elements passed in by each of the three arrays above.

rv ([ODBoolean](#)) - returns

A flag indication the success.

kODTrue	Successful completion
kODFalse	Error occurred

AddDetailsColumns (OS/2) - Remarks

This method should be called before the [DisplayView](#) method and can be called any number of time before calling that method. Additional column information is appended to the previous information already passed.

AddDetailsColumns (OS/2) - Topics

Class:

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

DisplayView (OS/2)

DisplayView (OS/2) - Syntax

This method opens a window containing the specified view.

```
#define INCL_ODVIEWEXTENSION
#define INCL_ODAPI
#include <os2.h>

ODPart      *rootpart;
ODULong     ulViewType;
ODBoolean    rv;

rv = DisplayView(rootpart, ulViewType);
```

DisplayView (OS/2) Parameter - rootpart

rootpart (ODPart *) - input
 The part to be the root part when the view is displayed.

DisplayView (OS/2) Parameter - ulViewType

ulViewType (ODULong) - input
 The view to be displayed. This parameter can be set to one of the following values:

OD_TREEVIEW	The embedded parts are displayed as a tree hierarchy.
OD_ICONVIEW	The embedded parts are displayed as icons.
OD_DETAILSVIEW	A list of embedded parts is displayed along with their properties information.

DisplayView (OS/2) Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indication the success.

kODTrue	Successful completion
kODFalse	Error occurred

DisplayView (OS/2) - Parameters

rootpart ([ODPart *](#)) - input
The part to be the root part when the view is displayed.

uiViewType ([ODULong](#)) - input
The view to be displayed. This parameter can be set to one of the following values:

OD_TREEVIEW	The embedded parts are displayed as a tree hierarchy.
OD_ICONVIEW	The embedded parts are displayed as icons.
OD_DETAILSVIEW	A list of embedded parts is displayed along with their properties information.

rv ([ODBoolean](#)) - returns
A flag indication the success.

kODTrue	Successful completion
kODFalse	Error occurred

DisplayView (OS/2) - Remarks

This method is called by the active part who owns this view extension. Part developers who subclass [ODViewExtension](#) and want to provide their own custom tree view need to override this method and pass the OpenDoc-defined view to the parent.

DisplayView (OS/2) - Topics

Class:
[ODViewExtension](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

InitViewExtension (OS/2)

InitViewExtension (OS/2) - Syntax

This method initializes the view extension object.

```
#define INCL_ODVIEWEXTENSION
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;

InitViewExtension(part);
```

InitViewExtension (OS/2) Parameter - part

part (ODPart *) - input
The part to be the owner of this object.

InitViewExtension (OS/2) - Return Value

None.

InitViewExtension (OS/2) - Parameters

part (ODPart *) - input
The part to be the owner of this object.

None.

InitViewExtension (OS/2) - Topics

Class:
ODViewExtension

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWindow

Class Definition File: WINDOW.IDL

Class Hierarchy

```
SOMObject
  ODOject
    ODORefCntObject
      ODOBaseWindow
        ODWindow
```

Description

An object of the ODWindow class is a wrapper for a platform-specific window structure.

Every window (except a modal dialog box) must be associated with an OpenDoc window object so that the part belonging to the root frame of the window, and its embedded parts, can receive events from the dispatcher. To make platform-specific calls, part editors can retrieve the platform-specific window from a window object. However, in most cases, the interface to the ODWindow class provides the capability you need for interacting with your platform-specific windows.

Your part creates a window object by calling the window-state object's [RegisterWindow](#) or [RegisterWindowForFrame](#) method.

When your part creates a window, it specifies whether the new window should be a root window. A document remains open as long as it has an open root window; the document shell closes the document when the last root window is closed. A root window is also called a document window. The initial window of a document is a root window; its root part is the root part of the document. Part windows, which display embedded parts, and the dialog boxes are no root windows. OpenDoc permits a single document to have multiple root windows as long as the root part provides a user interface to support them.

Your part should not maintain references to window objects for accessing OpenDoc windows because the document shell or the window-state object can close the window object and invalidate the reference. You should instead maintain window IDs, from the which the window object can be reconstructed.

Methods

The methods defined by the ODWindow class include:

Manipulating Windows

- [Close](#)
- [CloseAndRemove](#)
- [GetID](#)
- [GetPlatformWindow](#)
- [Hide](#)
- [Select](#)
- [Show](#)
- [Update](#)

Window Characteristics

- [IsActive](#)
- [IsFloating](#)
- [IsResizable](#)
- [IsRootWindow](#)
- [IsShown](#)
- [SetShouldSave](#)
- [SetShouldShowLinks](#)
- [ShouldSave](#)
- [ShouldShowLinks](#)

Manipulating Facets

- [GetFacetUnderPoint](#)
- [GetRootFacet](#)
- [Open](#)

Manipulating Frames

- [AcquireSourceFrame](#)
- [AdjustWindowShape](#)

- [GetRootFrame](#)
- [SetSourceFrame](#)

Overridden Methods

There are currently no methods overridden by the ODWindow class.

AcquireSourceFrame

AcquireSourceFrame - Syntax

This method returns a reference to the frame, if any, from which this window was opened.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = AcquireSourceFrame();
```

AcquireSourceFrame Return Value - rv

rv (ODFrame *) - returns

A reference to the frame from which this window was opened or KODNULL if the window does not have a source frame (for example, a root window).

AcquireSourceFrame - Parameters

rv (ODFrame *) - returns

A reference to the frame from which this window was opened or KODNULL if the window does not have a source frame (for example, a root window).

AcquireSourceFrame - Remarks

This method increments the reference count of the returned frame object. When you have finished using that frame object, you should call its [Release](#) method.

If your part class implements shared dialog windows (which are shared by all part objects of your class in a document), your part objects can use the source frame of the dialog window to find out which frame is currently associated with the dialog window.

It is possible to change the source frame later on by calling this window's [SetSourceFrame](#) method.

AcquireSourceFrame - Related Methods

Related Methods

- [ODWindow::SetSourceFrame](#)
-

AcquireSourceFrame - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

AdjustWindowShape

AdjustWindowShape - Syntax

This method reshapes the root frame to match the size of this window.

```
#define INCL_ODWINDOW  
#define INCL_ODAPI  
#include <os2.h>
```

```
AdjustWindowShape();
```

AdjustWindowShape - Return Value

None.

AdjustWindowShape - Parameters

None.

AdjustWindowShape - Remarks

Your part (the root part) calls this method after resizing the window.

AdjustWindowShape - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

Close

Close - Syntax

This method is called by the document shell or container application to close this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
Close();
```

Close - Return Value

None.

Close - Parameters

None.

Close - Remarks

OpenDoc calls this method when closing a document; parts typically call this window object's [CloseAndRemove](#) method instead of this method.

Close - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

CloseAndRemove

CloseAndRemove - Syntax

This method closes this window object and removes the root frame from its draft.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
CloseAndRemove();
```

CloseAndRemove - Return Value

None.

CloseAndRemove - Parameters

None.

CloseAndRemove - Remarks

This method closes this window object, releases this window object, deletes the root facet, and removes the root frame from the draft. Your part calls this method to remove auxiliary windows, such as palettes or part windows.

CloseAndRemove - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

GetFacetUnderPoint

GetFacetUnderPoint - Syntax

This method returns a reference to the facet of this window under the specified point.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODPoint      *aPoint;
ODFacet      *rv;

rv = GetFacetUnderPoint(aPoint);
```

GetFacetUnderPoint Parameter - aPoint

aPoint ([ODPoint *](#)) - input
The location to test, expressed in window coordinates.

GetFacetUnderPoint Return Value - rv

rv (ODFacet *) - returns
A reference to the facet under the specified point.

GetFacetUnderPoint - Parameters

aPoint ([ODPoint *](#)) - input
The location to test, expressed in window coordinates.

rv (ODFacet *) - returns
A reference to the facet under the specified point.

GetFacetUnderPoint - Remarks

OpenDoc calls this method for event dispatching; this method is not called by most parts.

If several nested facets are under the point, the innermost one is returned. If multiple overlapping frames are under the point, the unobscured (that is, the front-most) one is returned. The bundled and selected properties of frames and facets are respected.

GetFacetUnderPoint - Topics

Class:
[ODWindow](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetID

GetID - Syntax

This method returns the window ID for this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODID    rv;

rv = GetID();
```

GetID Return Value - rv

rv ([ODID](#)) - returns
The window ID for this window object.

GetID - Parameters

rv ([ODID](#)) - returns
The window ID for this window object.

GetID - Remarks

The window-state object assigns window IDs that are valid for the length of the session. You can use this method to get the window ID of this window object when it is created, and then pass that ID to the window-state object's [AcquireWindow](#) method for subsequent access to this window object.

GetID - Related Methods

Related Methods

- [ODWindowState::AcquireWindow](#)

GetID - Topics

Class:
ODWindow

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetPlatformWindow

GetPlatformWindow - Syntax

This method returns the platform-specific window for this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindow    rv;

rv = GetPlatformWindow();
```

GetPlatformWindow Return Value - rv

rv ([ODPlatformWindow](#)) - returns
The platform-specific window for this window object.

GetPlatformWindow - Parameters

rv ([ODPlatformWindow](#)) - returns
The platform-specific window for this window object.

GetPlatformWindow - Topics

Class:
ODWindow

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetRootFacet

GetRootFacet - Syntax

This method returns a reference to the root facet of this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = GetRootFacet();
```

GetRootFacet Return Value - rv

rv (ODFacet *) - returns
A reference to the root facet of this window object.

GetRootFacet - Parameters

rv (ODFacet *) - returns
A reference to the root facet of this window object.

GetRootFacet - Topics

Class:
ODWindow

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

GetRootFrame

GetRootFrame - Syntax

This method returns a reference to the root frame of this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = GetRootFrame();
```

GetRootFrame Return Value - rv

rv (ODFrame *) - returns
A reference to the root frame of this window object.

GetRootFrame - Parameters

rv (ODFrame *) - returns
A reference to the root frame of this window object.

GetRootFrame - Remarks

This method does not increment the reference count of the returned frame. For that reason, if you cache the returned frame, you should call its [Acquire](#) method to increment its reference count and then call its [Release](#) method when you have finished using it.

GetRootFrame - Topics

Class:
ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

Hide

Hide - Syntax

This method makes this window invisible.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
Hide();
```

Hide - Return Value

None.

Hide - Parameters

None.

Hide - Related Methods

Related Methods

- [ODWindow::Show](#)
-

Hide - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Related Methods](#)

IsActive

IsActive - Syntax

This method indicates whether this window is active.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsActive();
```

IsActive Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is active.

kODTrue

This window is active; it is either the frontmost, nonfloating window or a floating window.

kODFalse

This window is not active.

IsActive - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is active.

kODTrue

This window is active; it is either the frontmost, nonfloating window or a floating window.

kODFalse

This window is not active.

IsActive - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

IsFloating

IsFloating - Syntax

This method indicates whether this window is a floating window.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsFloating();
```

IsFloating Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is a floating window.

kODTrue	The window is a floating window.
kODFalse	The window is not a floating window.

IsFloating - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is a floating window.

kODTrue	The window is a floating window.
kODFalse	The window is not a floating window.

IsFloating - Topics

Class:
ODWindow

Select an item:
[Syntax](#)
[Parameters](#)

IsResizable

IsResizable - Syntax

This method indicates whether this window is resizable.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsResizable();
```

IsResizable Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is resizable.

kODTrue	This window is resizable.
kODFalse	This window is not resizable.

IsResizable - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is resizable.

kODTrue	This window is resizable.
kODFalse	This window is not resizable.

IsResizable - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

IsRootWindow

IsRootWindow - Syntax

This method indicates whether this window object is a root window.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsRootWindow();
```

IsRootWindow Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window object is a root window.

kODTrue This window object is a root window.

kODFalse This window object is not a root window.

IsRootWindow - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this window object is a root window.

kODTrue This window object is a root window.

kODFalse This window object is not a root window.

IsRootWindow - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

IsShown

IsShown - Syntax

This method indicates whether this window is currently visible.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsShown();
```

IsShown Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is currently visible.

kODTrue

This window is currently visible.

kODFalse

This window is not currently visible.

IsShown - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether this window is currently visible.

kODTrue

This window is currently visible.

kODFalse

This window is not currently visible.

IsShown - Related Methods

Related Methods

- [ODWindow::Hide](#)
- [ODWindow::Show](#)

IsShown - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Related Methods](#)

Open

Open - Syntax

This method creates a facet hierarchy in this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
Open();
```

Open - Return Value

None.

Open - Parameters

None.

Open - Remarks

Your part calls this method when first opening a window. This method does not make the window visible or change window ordering; for those operations, your part calls this window object's [Show](#) and [Select](#) methods instead.

Open - Related Methods

Related Methods

- [ODWindow::Select](#)
 - [ODWindow::Show](#)
-

Open - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

Select

Select - Syntax

This method makes this window the front-most window and activates it.

```
#define INCL_ODWINDOW  
#define INCL_ODAPI  
#include <os2.h>
```

```
Select();
```

Select - Return Value

None.

Select - Parameters

None.

Select - Remarks

Your part calls this method when first opening or when later activating a window. This method changes window ordering.

Select - Related Methods

Related Methods

- [ODWindow::Show](#)
-

Select - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

SetShouldSave

SetShouldSave - Syntax

This method specifies whether this window object should be saved in its draft.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODBoolean    shouldSave;

SetShouldSave(shouldSave);
```

SetShouldSave Parameter - shouldSave

shouldSave (ODBoolean) - input

A flag indicating whether the window should be saved in the draft.

kODTrue	The window should be saved.
kODFalse	The window should not be saved.

SetShouldSave - Return Value

None.

SetShouldSave - Parameters

shouldSave (ODBoolean) - input

A flag indicating whether the window should be saved in the draft.

kODTrue	The window should be saved.
kODFalse	The window should not be saved.

None.

SetShouldSave - Remarks

This property should generally be set to kODTrue for root windows and kODFalse for other windows.

SetShouldSave - Related Methods

Related Methods

- [ODWindow::ShouldSave](#)

SetShouldSave - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

SetShouldShowLinks

SetShouldShowLinks - Syntax

This method specifies whether links should be highlighted in this window.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    shouldShowLinks;
```

```
SetShouldShowLinks (shouldShowLinks);
```

SetShouldShowLinks Parameter - shouldShowLinks

shouldShowLinks ([ODBoolean](#)) - input

A flag indicating whether the links should be highlighted in this window.

kODTrue

The links should be highlighted in this window.

kODFalse

The links should not be highlighted in this window.

SetShouldShowLinks - Return Value

None.

SetShouldShowLinks - Parameters

shouldShowLinks ([ODBoolean](#)) - input

A flag indicating whether the links should be highlighted in this window.

kODTrue

The links should be highlighted in this window.

kODFalse

The links should not be highlighted in this window.

None.

SetShouldShowLinks - Related Methods

Related Methods

- [ODWindow::ShouldShowLinks](#)
-

SetShouldShowLinks - Topics

Class:

[ODWindow](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

SetSourceFrame

SetSourceFrame - Syntax

This method sets the source frame of this window object.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

SetSourceFrame(frame);
```

SetSourceFrame Parameter - frame

frame (ODFrame *) - input
A reference to the new source frame.

SetSourceFrame - Return Value

None.

SetSourceFrame - Parameters

frame (ODFrame *) - input
A reference to the new source frame.

None.

SetSourceFrame - Related Methods

Related Methods

- [ODWindow::AcquireSourceFrame](#)

SetSourceFrame - Topics

Class:
ODWindow

Select an item:

[Syntax](#)
[Parameters](#)

ShouldSave

ShouldSave - Syntax

This method indicates whether this window object is saved in its draft.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = ShouldSave();
```

ShouldSave Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether this window object is saved in its draft.

kODTrue	This window object is saved.
kODFalse	This window object is not saved.

ShouldSave - Parameters

rv ([ODBoolean](#)) - returns
A flag indicating whether this window object is saved in its draft.

kODTrue	This window object is saved.
kODFalse	This window object is not saved.

ShouldSave - Related Methods

Related Methods

ShouldSave - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

ShouldShowLinks

ShouldShowLinks - Syntax

This method indicates whether links are highlighted in this window.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = ShouldShowLinks();
```

ShouldShowLinks Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether links are highlighted in this window.

kODTrue

The links are highlighted in this window.

kODFalse

The links are not highlighted in this window.

ShouldShowLinks - Parameters

rv ([ODBoolean](#)) - returns

A flag indicating whether links are highlighted in this window.

kODTrue

The links are highlighted in this window.

kODFalse

The links are not highlighted in this window.

ShouldShowLinks - Related Methods

Related Methods

- [ODWindow::SetShouldShowLinks](#)

ShouldShowLinks - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

Show

Show - Syntax

This method makes this window visible.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
Show();
```

Show - Return Value

None.

Show - Parameters

None.

Show - Remarks

Your part calls this method when first opening a window and when making a window visible after having called its [Hide](#) method. This method does not change window ordering.

Show - Related Methods

Related Methods

- [ODWindow::Hide](#)
 - [ODWindow::Select](#)
-

Show - Topics

Class:

ODWindow

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

Update (OS/2)

Update (OS/2) - Syntax

This method forces immediate updating of this window, rather than waiting for an update event.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODEventData *theEvent;
```

```
Update(theEvent);
```

Update (OS/2) Return Value - theEvent

theEvent (ODEventData *) - returns

The event that just occurred within your part.

Update (OS/2) - Return Value

None.

Update (OS/2) - Parameters

theEvent (ODEventData *) - returns

The event that just occurred within your part.

None.

Update (OS/2) - Remarks

When an update event occurs that involves a facet of your part, OpenDoc calls this method, which in turn calls its facet's [Update](#) method, which then calls the [Draw](#) method associated with the facet's part. Your part might also call this method to force updating when it does not happen automatically, for instance, when there is a mouse-down event and you cannot wait for an update event.

Update (OS/2) - Related Methods

Related Methods

- [ODFacet::Update](#)
- [ODPart::Draw](#)

Update (OS/2) - Topics

Class:

ODWindow

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

ODWindowIterator

Class Definition File: WINITER.IDL

Class Hierarchy

SOMObject

ODObject

ODWindowIterator

Description

An object of the ODWindowIterator class provides access to all windows of the window-state object.

You use a window iterator to apply an operation to all windows of all open drafts of the current session's document. For example, a root part might use a window iterator to tile all the currently open windows. A window iterator maintains a reference to its window-state object and to the current window object. The internal list of windows in the window-state object is ordered by creation time and is not related to front-to-back ordering of the windows.

Your part creates a window iterator object by calling the window-state object's [CreateWindowIterator](#) method, which returns a reference to a window iterator object.

While you are using a window iterator, you should not modify the list of opened windows. You must postpone adding windows to or removing windows from the list of open windows until after you have deleted the iterator.

For more information related to the window-state object, see the class description for [ODWindowState](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

Methods

The methods defined by the ODWindowIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Last](#)
- [Next](#)
- [Previous](#)

Overridden Methods

There are currently no methods overridden by the ODWindowIterator class.

First

First - Syntax

This method begins the iteration and returns a reference to the first window in the window state.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = First();
```

First Return Value - rv

rv (ODWindow *) - returns
A reference to the first window in the iteration sequence or KODNULL for an empty window state.

First - Parameters

rv (ODWindow *) - returns
A reference to the first window in the iteration sequence or KODNULL for an empty window state.

First - Remarks

If you are iterating from the first window to the last, your part must call this method before calling this window iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

The [Next](#) method is called to step through the window list from first to last.

This method does not increment the reference count of the returned window object.

First - Exception Handling

KODErrIteratorOutOfSync

The list of open windows was modified while the iteration was in progress.

First - Topics

Class:
ODWindowIterator

Select an item:

IsNotComplete

IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsNotComplete();
```

IsNotComplete Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

IsNotComplete - Parameters

rv ([ODBoolean](#)) - returns
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

IsNotComplete - Remarks

Your part calls this method to test whether more windows remain in the window state. This method returns kODTrue if the preceeding call to the [First](#), [Last](#), [Next](#), or [Previous](#) method found a window. This method returns kODFalse when you have examined all the windows (that is, when the previous call to [First](#), [Last](#), [Next](#), or [Previous](#) returned kODNULL).

IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The list of open windows was modified while the iteration was in progress.

IsNotComplete - Topics

Class:

ODWindowIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

Last

Last - Syntax

This method begins the iteration and returns a reference to the last window in the window state.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindow      *rv;
```

```
rv = Last();
```

Last Return Value - rv

rv (ODWindow *) - returns

A reference to the last window in the window state.

Last - Parameters

rv (ODWindow *) - returns
A reference to the last window in the window state.

Last - Remarks

If you are iterating from the last window to the first, your part must call this method before calling this window iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

The [Previous](#) method is called to step through the window list from last to first.

This method does not increment the reference count of the returned window object.

Last - Exception Handling

kODErrIteratorOutOfSync

The list of open windows was modified while the iteration was in progress.

Last - Topics

Class:
ODWindowIterator

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

Next

Next - Syntax

This method returns a reference to the next window in the window state.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = Next();
```

Next Return Value - rv

rv (ODWindow *) - returns
A reference to the next window in the window state or KODNULL if you have reached the last window.

Next - Parameters

rv (ODWindow *) - returns
A reference to the next window in the window state or KODNULL if you have reached the last window.

Next - Remarks

If your part calls this method before calling this window iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

This method does not increment the reference count of the returned window object.

Next - Exception Handling

KODErrIteratorOutOfSync

The list of open windows was modified while the iteration was in progress.

Next - Topics

Class:
ODWindowIterator

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)

Previous

Previous - Syntax

This method returns a reference to the previous window in the window state.

```
#define INCL_ODWINDOW
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindow      *rv;

rv = Previous();
```

Previous Return Value - rv

rv (ODWindow *) - returns

A reference to the previous window in the window state or KODNULL if you have reached the first window.

Previous - Parameters

rv (ODWindow *) - returns

A reference to the previous window in the window state or KODNULL if you have reached the first window.

Previous - Remarks

If your part calls this method before calling this window iterator's [Last](#) method to begin the iteration, this method works the same as calling the [Last](#) method.

This method does not increment the reference count of the returned window object.

Previous - Exception Handling

Previous - Topics

Class:

ODWindowIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

ODWindowState

Class Definition File: WINSTAT.IDL

Class Hierarchy

SOMObject

ODObject

ODBaseWindowState

ODWindowState

Description

An object of the ODWindowState class maintains a list of all the open window objects for all open drafts in an OpenDoc session and references to the base menu bar and the current menu bar.

When a document is opened, the session object creates a single window-state object. All parts of that document share the window-state object; you can obtain a reference to it by calling the session object's [GetWindowState](#) method. The document shell and dispatcher use the window-state object to pass events to parts so the parts can activate themselves, handle user input, and adjust their menus as necessary. The document shell may manage more than one open document. Typically, however, there is only one open document in a session, but multiple drafts of the document may be open. A part may be displayed in any number of frames, in any window of a document. The dispatcher passes events to the correct part, without regards to which window encloses the active frame and how many other frames the part has.

Your part accesses the window-state object to create new windows, to access a particular window, and to access the base menu-bar object.

For more information related to window objects, see the class description for [ODWindow](#).

Methods

The methods defined by the ODWindowState class include:

Creating Windows

- [RegisterWindow](#)
- [RegisterWindowForFrame](#)

Manipulating Windows

- [AcquireActiveWindow](#)
- [AcquireFrontWindow](#)
- [AcquireFrontFloatingWindow](#)
- [AcquireFrontRootWindow](#)
- [AcquireODWindow](#)
- [AcquireWindow](#)
- [CloseWindows](#)
- [CreateWindowIterator](#)
- [Externalize](#)
- [Internalize](#)
- [SetDefaultWindowTitles](#)
- [OpenWindows](#)

Window Characteristics

- [IsODWindow](#)
- [GetRootWindowCount](#)
- [GetTotalRootWindowCount](#)
- [GetWindowCount](#)

Manipulating Menu Bars

- [AdjustPartMenus](#)
- [AcquireBaseMenuBar](#)
- [AcquireCurrentMenuBar](#)
- [CopyBaseMenuBar](#)
- [CopyBasePopup](#)
- [CreateMenuBar](#)
- [SetBaseMenuBar](#)
- [SetBasePopup](#)

Creating Facets and Canvases

- [CreateCanvas](#)
- [CreateFacet](#)

Overridden Methods

There are currently no methods overridden by the ODWindowState class.

AcquireActiveWindow

AcquireActiveWindow - Syntax

This method returns a reference to the front-most, nonfloating window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = AcquireActiveWindow();
```

AcquireActiveWindow Return Value - rv

rv (ODWindow *) - returns

A reference to the frontmost nonfloating window or KODNULL if the frontmost nonfloating window is not an OpenDoc window.

AcquireActiveWindow - Parameters

rv (ODWindow *) - returns

A reference to the frontmost nonfloating window or KODNULL if the frontmost nonfloating window is not an OpenDoc window.

AcquireActiveWindow - Remarks

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

AcquireActiveWindow - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

AcquireBaseMenuBar

AcquireBaseMenuBar - Syntax

This method is called by the document shell to retrieve a reference to the base menu bar.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuBar      *rv;
```

```
rv = AcquireBaseMenuBar();
```

AcquireBaseMenuBar Return Value - rv

rv (ODMenuBar *) - returns

The base menu bar.

AcquireBaseMenuBar - Parameters

rv (ODMenuBar *) - returns
The base menu bar.

AcquireBaseMenuBar - Remarks

The document shell calls this method. Your part typically calls this window-state object's [CopyBaseMenuBar](#) method instead of this method.

This method increments the reference count of the returned menu-bar object. When the caller has finished using that menu-bar object, it should call the menu bar's [Release](#) method.

AcquireBaseMenuBar - Related Methods

Related Methods

- [ODWindowState::CopyBaseMenuBar](#)
 - [ODWindowState::SetBasePopup](#)
-

AcquireBaseMenuBar - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

AcquireCurrentMenuBar

AcquireCurrentMenuBar - Syntax

This method is called by the document shell to retrieve a reference to the menu bar being displayed by the document shell.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuBar      *rv;
```

```
rv = AcquireCurrentMenuBar();
```

AcquireCurrentMenuBar Return Value - rv

rv (ODMenuBar *) - returns
A reference to the menu bar currently being displayed.

AcquireCurrentMenuBar - Parameters

rv (ODMenuBar *) - returns
A reference to the menu bar currently being displayed.

AcquireCurrentMenuBar - Remarks

The document shell calls this method. Your part typically calls this window-state object's [CopyBaseMenuBar](#) method instead of this method.

This method increments the reference count of the returned menu-bar object. When the caller has finished using that menu-bar object, it should call the menu bar's [Release](#) method.

AcquireCurrentMenuBar - Related Methods

Related Methods

- [ODWindowState::CopyBaseMenuBar](#)

AcquireCurrentMenuBar - Topics

Class:
[ODWindowState](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

AcquireFrontFloatingWindow (OS/2)

AcquireFrontFloatingWindow (OS/2) - Syntax

This method returns a reference to the front-most, floating window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = AcquireFrontFloatingWindow();
```

AcquireFrontFloatingWindow (OS/2) Return Value - rv

rv (ODWindow *) - returns

A reference to the frontmost floating window or KODNULL if there are no floating OpenDoc windows.

AcquireFrontFloatingWindow (OS/2) - Parameters

rv (ODWindow *) - returns

A reference to the frontmost floating window or KODNULL if there are no floating OpenDoc windows.

AcquireFrontFloatingWindow (OS/2) - Remarks

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

AcquireFrontFloatingWindow (OS/2) - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

AcquireFrontRootWindow (OS/2)

AcquireFrontRootWindow (OS/2) - Syntax

This method returns a reference to the frontmost (nonfloating) root window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = AcquireFrontRootWindow();
```

AcquireFrontRootWindow (OS/2) Return Value - rv

rv (ODWindow *) - returns
A reference to the frontmost (nonfloating) root window.

AcquireFrontRootWindow (OS/2) - Parameters

rv (ODWindow *) - returns
A reference to the frontmost (nonfloating) root window.

AcquireFrontRootWindow (OS/2) - Remarks

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

AcquireFrontRootWindow (OS/2) - Topics

Class:
ODWindowState

Select an item:
[Syntax](#)
[Parameters](#)

AcquireFrontWindow (OS/2)

AcquireFrontWindow (OS/2) - Syntax

This method returns a reference to the frontmost window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = AcquireFrontWindow();
```

AcquireFrontWindow (OS/2) Return Value - rv

rv (ODWindow *) - returns
A reference to the frontmost window or KODNULL if the frontmost window is not an OpenDoc window.

AcquireFrontWindow (OS/2) - Parameters

rv (ODWindow *) - returns
A reference to the frontmost window or KODNULL if the frontmost window is not an OpenDoc window.

AcquireFrontWindow (OS/2) - Remarks

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

AcquireFrontWindow (OS/2) - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

AcquireODWindow

AcquireODWindow - Syntax

This method returns a reference to the window object corresponding to the specified platform-specific window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindow  aWindow;
ODWindow          *rv;

rv = AcquireODWindow(aWindow);
```

AcquireODWindow Parameter - aWindow

aWindow ([ODPlatformWindow](#)) - input
A platform-specific window.

AcquireODWindow Return Value - rv

rv (ODWindow *) - returns
A reference to the window object corresponding to the specified platform-specific window or KODNULL if the window is not an OpenDoc window.

AcquireODWindow - Parameters

aWindow ([ODPlatformWindow](#)) - input
A platform-specific window.

rv (ODWindow *) - returns

A reference to the window object corresponding to the specified platform-specific window or KODNULL if the window is not an OpenDoc window.

AcquireODWindow - Remarks

OpenDoc calls this method.

This method increments the reference count of the returned window object. When the caller has finished using that window object, it should call the window's [Release](#) method.

AcquireODWindow - Related Methods

Related Methods

- [ODWindowState::IsODWindow](#)
-

AcquireODWindow - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

AcquireWindow

AcquireWindow - Syntax

This method returns a reference to the window object with the specified ID.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODID      id;
ODWindow  *rv;

rv = AcquireWindow(id);
```

AcquireWindow Parameter - id

id ([ODID](#)) - input
The window ID for the window object.

AcquireWindow Return Value - rv

rv (ODWindow *) - returns
A reference to the window object with the specified ID, or KODNULL if the window has been deleted or does not exist.

AcquireWindow - Parameters

id ([ODID](#)) - input
The window ID for the window object.

rv (ODWindow *) - returns
A reference to the window object with the specified ID, or KODNULL if the window has been deleted or does not exist.

AcquireWindow - Remarks

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

AcquireWindow - Related Methods

Related Methods

- [ODWindow::GetID](#)
-

AcquireWindow - Topics

Class:
ODWindowState

Select an item:
[Syntax](#)
[Parameters](#)

AdjustPartMenus

AdjustPartMenus - Syntax

This method calls both the root part and the part with the menu focus to adjust the displayed menu.

```
#define INCL_ODWINDOWSTATE  
#define INCL_ODAPI  
#include <os2.h>
```

```
AdjustPartMenus();
```

AdjustPartMenus - Return Value

None.

AdjustPartMenus - Parameters

None.

AdjustPartMenus - Remarks

OpenDoc calls this method when a mouse-down event occurs in the menu bar. This method in turn calls the [AdjustMenus](#) method for both the root part and the part with the menu focus, so the parts can enable or disable menu items as necessary. If the root part has the menu focus, then OpenDoc only calls the [AdjustMenus](#) method once.

AdjustPartMenus - Related Methods

Related Methods

- [ODPart::AdjustMenus](#)

AdjustPartMenus - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

CloseWindows

CloseWindows - Syntax

This method is called by the document shell or container application to close all windows belonging to the specified draft.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;

CloseWindows(draft);
```

CloseWindows Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

CloseWindows - Return Value

None.

CloseWindows - Parameters

draft (ODDraft *) - input
A reference to the open draft object.

None.

CloseWindows - Remarks

The document shell calls this method when closing a draft.

CloseWindows - Related Methods

Related Methods

- [ODWindowState::OpenWindows](#)
-

CloseWindows - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

CopyBaseMenuBar

CopyBaseMenuBar - Syntax

This method copies the base menu-bar object.

```
#define INCL_ODWINDOWSTATE
```

```
#define INCL_ODAPI
#include <os2.h>

ODMenuBar      *rv;

rv = CopyBaseMenuBar();
```

CopyBaseMenuBar Return Value - rv

rv (ODMenuBar *) - returns
A reference to the newly created copy of the base menu-bar object.

CopyBaseMenuBar - Parameters

rv (ODMenuBar *) - returns
A reference to the newly created copy of the base menu-bar object.

CopyBaseMenuBar - Remarks

Your part calls this method to create a menu-bar object to which it can add its own menus.

This method initializes the reference count of the returned menu-bar object. When you have finished using that menu-bar object, you should call its [Release](#) method.

CopyBaseMenuBar - Related Methods

Related Methods

- [ODMenuBar::Copy](#)
- [ODWindowState::SetBaseMenuBar](#)

CopyBaseMenuBar - Topics

Class:
ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

CopyBasePopup (OS/2)

CopyBasePopup (OS/2) - Syntax

This method copies the pop-up menu object.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPopup      *rv;

rv = CopyBasePopup();
```

CopyBasePopup (OS/2) Return Value - rv

rv (ODPopup *) - returns
A reference to a copy of the pop-up menu object.

CopyBasePopup (OS/2) - Parameters

rv (ODPopup *) - returns
A reference to a copy of the pop-up menu object.

CopyBasePopup (OS/2) - Remarks

Your part editor calls this method to create a pop-up menu object to which it can add its own menu items. This method in turn calls its pop-up menu object's [CopyPopup](#) method.

This method increments the reference count of the returned pop-up menu object. When you have finished using that pop-up menu object, you should call its [Release](#) method.

CopyBasePopup (OS/2) - Related Methods

Related Methods

CopyBasePopup (OS/2) - Topics

Class:

[ODWindowState](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

CreateCanvas

CreateCanvas - Syntax

This method creates a canvas object.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem      graphicsSystem;
ODPlatformCanvas      *platformCanvas;
ODBoolean              isDynamic;
ODBoolean              isOffscreen;
ODCanvas               *rv;

rv = CreateCanvas(graphicsSystem, platformCanvas,
                  isDynamic, isOffscreen);
```

CreateCanvas Parameter - graphicsSystem

graphicsSystem ([ODGraphicsSystem](#)) - input

The graphics system to be used for the canvas. Valid graphics systems for this parameter are platform-dependent. On OS/2, this parameter should always be set to KODGPI.

CreateCanvas Parameter - platformCanvas

platformCanvas ([ODPlatformCanvas *](#)) - input

The graphics-system-specific drawing structure to be assigned to the canvas or kODNULL if there is no drawing structure associated with the specified graphics system. Valid values for platformCanvas are graphics-system-dependent.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure.

CreateCanvas Parameter - isDynamic

isDynamic ([ODBoolean](#)) - input

A flag indicating whether the canvas is to be dynamic.

kODTrue

The canvas is to be dynamic.

kODFalse

The canvas is to be static.

CreateCanvas Parameter - isOffscreen

isOffscreen ([ODBoolean](#)) - input

A flag indicating whether the canvas is to be offscreen.

kODTrue

The canvas is to be offscreen.

kODFalse

The canvas is to be onscreen.

CreateCanvas Return Value - rv

rv ([ODCanvas *](#)) - returns

A reference to the newly created canvas object.

CreateCanvas - Parameters

graphicsSystem ([ODGraphicsSystem](#)) - input

The graphics system to be used for the canvas. Valid graphics systems for this parameter are platform-dependent. On OS/2, this parameter should always be set to kODGPI.

platformCanvas ([ODPlatformCanvas *](#)) - input

The graphics-system-specific drawing structure to be assigned to the canvas or kODNULL if there is no drawing structure associated with the specified graphics system. Valid values for platformCanvas are graphics-system-dependent.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure.

isDynamic ([ODBoolean](#)) - input

A flag indicating whether the canvas is to be dynamic.

kODTrue

kODFalse	The canvas is to be dynamic.
	The canvas is to be static.

isOffscreen ([ODBoolean](#)) - input
A flag indicating whether the canvas is to be offscreen.

kODTrue	The canvas is to be offscreen.
kODFalse	The canvas is to be onscreen.

rv (ODCanvas *) - returns
A reference to the newly created canvas object.

CreateCanvas - Remarks

Your part calls this method to create a canvas object that will not be attached to any facet. To create a canvas to attach to a particular facet, you should call that facet's [CreateCanvas](#) method instead of this method.

CreateCanvas - Related Methods

Related Methods

- [ODFacet::CreateCanvas](#)

CreateCanvas - Topics

Class:
ODWindowState

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

CreateFacet

CreateFacet - Syntax

This method creates a facet object.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODShape      *clipShape;
ODTransform  *externalTransform;
ODCanvas     *canvas;
ODCanvas     *biasCanvas;
ODFacet      *rv;

rv = CreateFacet(frame, clipShape, externalTransform,
                 canvas, biasCanvas);
```

CreateFacet Parameter - frame

frame (ODFrame *) - input
A reference to the frame for the facet.

CreateFacet Parameter - clipShape

clipShape (ODShape *) - input
A reference to the initial clip shape for the facet.

CreateFacet Parameter - externalTransform

externalTransform (ODTransform *) - input
A reference to the initial external transform for the facet.

CreateFacet Parameter - canvas

canvas (ODCanvas *) - input
A reference to the canvas the facet should draw to or KODNULL if identical to the canvas associated with the containing facet.

CreateFacet Parameter - biasCanvas

biasCanvas (ODCanvas *) - input
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in the standard platform-normal coordinates.

CreateFacet Return Value - rv

rv (ODFacet *) - returns
A reference to the newly created facet object.

CreateFacet - Parameters

frame (ODFrame *) - input
A reference to the frame for the facet.

clipShape (ODShape *) - input
A reference to the initial clip shape for the facet.

externalTransform (ODTransform *) - input
A reference to the initial external transform for the facet.

canvas (ODCanvas *) - input
A reference to the canvas the facet should draw to or kODNULL if identical to the canvas associated with the containing facet.

biasCanvas (ODCanvas *) - input
A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in the standard platform-normal coordinates.

rv (ODFacet *) - returns
A reference to the newly created facet object.

CreateFacet - Remarks

Your part calls this method to create a root facet (for example, for printing). The frame is defined for the lifetime of the facet object; once set, it cannot be changed.

To create a facet object for a visible embedded frame, your part should call its own display facet's [CreateEmbeddedFacet](#) method instead of this method.

CreateFacet - Related Methods

Related Methods

- [ODFacet::CreateEmbeddedFacet](#)
-

CreateFacet - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

CreateMenuBar

CreateMenuBar - Syntax

This method is called by the document shell to create and initialize a menu-bar object.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPlatformMenuBar    menuBar;
ODMenuBar            *rv;

rv = CreateMenuBar(menuBar);
```

CreateMenuBar Parameter - menuBar

menuBar ([ODPlatformMenuBar](#)) - input
A platform-specific menu-bar.

CreateMenuBar Return Value - rv

rv (ODMenuBar *) - returns
A reference to the newly created menu-bar object.

CreateMenuBar - Parameters

menuBar ([ODPlatformMenuBar](#)) - input
A platform-specific menu-bar.

rv (ODMenuBar *) - returns

A reference to the newly created menu-bar object.

CreateMenuBar - Remarks

The document shell calls this method. Your part typically calls this window-state object's [CopyBaseMenuBar](#) method instead of this method.

This method initializes the reference count of the returned menu bar. When the caller has finished using that menu bar, it should call the menu bar's [Release](#) method.

CreateMenuBar - Related Methods

Related Methods

- [ODWindowState::CopyBaseMenuBar](#)
-

CreateMenuBar - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

CreateWindowIterator

CreateWindowIterator - Syntax

This method creates a window iterator for the windows (of all drafts) in this window-state object.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODWindowIterator      *rv;

rv = CreateWindowIterator();
```

CreateWindowIterator Return Value - rv

rv (ODWindowIterator *) - returns
A reference to the newly created window iterator.

CreateWindowIterator - Parameters

rv (ODWindowIterator *) - returns
A reference to the newly created window iterator.

CreateWindowIterator - Remarks

Your part calls this method if it needs to apply an operation to all windows of a window-state object. For example, a root part might use a window iterator to tile all the currently open windows.

While you are using the returned winndow iterator, you must not call any methods that create or delete windows.

You must delete the returned window iterator when it is no longer needed.

CreateWindowIterator - Topics

Class:
ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

Externalize

Externalize - Syntax

This method is called by the document shell to write to persistent storage the window properties of windows of the specified draft and saves their frames as the root frames of the draft.

```
#define INCL_ODWINDOWSTATE
```

```
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;

Externalize(draft);
```

Externalize Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

Externalize - Return Value

None.

Externalize - Parameters

draft (ODDraft *) - input
A reference to the open draft object.

None.

Externalize - Remarks

The document shell calls this method when saving a draft. This method saves window properties for those windows of the specified draft that should be saved—that is, for those windows whose [ShouldSave](#) method returns kODTrue.

Externalize - Related Methods

Related Methods

- [ODWindow::ShouldSave](#)
- [ODWindowState::Internalize](#)

Externalize - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

GetRootWindowCount

GetRootWindowCount - Syntax

This method returns the number of root windows belonging to the specified draft.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;
ODUShort      rv;

rv = GetRootWindowCount(draft);
```

GetRootWindowCount Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

GetRootWindowCount Return Value - rv

rv (ODUShort) - returns
The number of root windows belonging to the specified draft.

GetRootWindowCount - Parameters

draft (ODDraft *) - input

A reference to the open draft object.

rv ([ODUShort](#)) - returns

The number of root windows belonging to the specified draft.

GetRootWindowCount - Remarks

The document shell calls this method when closing a window to determine when to close the draft; the draft is closed when its last root window is closed.

GetRootWindowCount - Related Methods

Related Methods

- [ODWindowState::GetTotalRootWindowCount](#)

GetRootWindowCount - Topics

Class:

[ODWindowState](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

GetTotalRootWindowCount

GetTotalRootWindowCount - Syntax

This method returns the total number of root windows of all open drafts.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODUShort    rv;

rv = GetTotalRootWindowCount();
```

GetTotalRootWindowCount Return Value - rv

rv ([ODUShort](#)) - returns
The number of root windows of all drafts.

GetTotalRootWindowCount - Parameters

rv ([ODUShort](#)) - returns
The number of root windows of all drafts.

GetTotalRootWindowCount - Related Methods

Related Methods

- [ODWindowState::GetRootWindowCount](#)
-

GetTotalRootWindowCount - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Related Methods](#)

GetWindowCount

GetWindowCount - Syntax

This method returns the number of windows (of all open drafts) in this window-state object.

```
#define INCL_ODWINDOWSTATE  
#define INCL_ODAPI
```

```
#include <os2.h>

ODUShort    rc;

rc = GetWindowCount();
```

GetWindowCount Return Value - rc

rc ([ODUShort](#)) - returns
The total number of windows in this window-state object.

GetWindowCount - Parameters

rc ([ODUShort](#)) - returns
The total number of windows in this window-state object.

GetWindowCount - Topics

Class:
ODWindowState

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

Internalize

Internalize - Syntax

This method is called by the document shell to read into memory all root frames of the specified draft, causing their parts to open the windows.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODDraft    *draft;

Internalize(draft);
```

Internalize Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

Internalize - Return Value

None.

Internalize - Parameters

draft (ODDraft *) - input
A reference to the open draft object.

None.

Internalize - Remarks

The document shell calls this method when opening a draft. After reading the root frames, this method reads into memory the part associated with each root frame, then passes that root frame as the parameter to its part's [Open](#) method. The [Open](#) method in turn creates the root (document) window for the root frame.

Internalize - Exception Handling

kODErrOutOfMemory

There is not enough memory to read in the data.

Internalize - Related Methods

Related Methods

- [ODPart::Open](#)
- [ODWindowState::Externalize](#)

Internalize - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

IsODWindow

IsODWindow - Syntax

This method indicates whether this window-state object has a window with the specified platform-specific window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindow    aWindow;
ODBoolean           rv;

rv = IsODWindow(aWindow);
```

IsODWindow Parameter - aWindow

aWindow ([ODPlatformWindow](#)) - input

A platform-specific window structure.

IsODWindow Return Value - rv

rv ([ODBoolean](#)) - returns

A flag indicating whether this window-state object has a window with the specified platform-specific window.

kODTrue

This window-state object has a window with the specified platform-specific window.

kODFalse

This window-state object does not have a window with the specified platform-specific window.

IsODWindow - Parameters

aWindow ([ODPlatformWindow](#)) - input
A platform-specific window structure.

rv ([ODBoolean](#)) - returns
A flag indicating whether this window-state object has a window with the specified platform-specific window.

kODTrue	This window-state object has a window with the specified platform-specific window.
kODFalse	This window-state object does not have a window with the specified platform-specific window.

IsODWindow - Remarks

OpenDoc calls this method.

IsODWindow - Related Methods

Related Methods

- [ODWindowState::AcquireODWindow](#)
-

IsODWindow - Topics

Class:
[ODWindowState](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Related Methods](#)

OpenWindows

OpenWindows - Syntax

This method is called by the document shell or container application to open all windows belonging to the specified draft.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;

OpenWindows(draft);
```

OpenWindows Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

OpenWindows - Return Value

None.

OpenWindows - Parameters

draft (ODDraft *) - input
A reference to the open draft object.

None.

OpenWindows - Remarks

If the draft is already open, this method brings its window to the front.

OpenWindows - Exception Handling

kODErrOutOfMemory

There is not enough memory to open all the windows.

OpenWindows - Related Methods

Related Methods

- [ODWindowState::CloseWindows](#)
-

OpenWindows - Topics

Class:

[ODWindowState](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

RegisterWindow

RegisterWindow - Syntax

This method creates an OpenDoc window object and root frame for the specified platform-specific window.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindow    newWindow;
ODType              frameType;
ODBoolean            isRootWindow;
ODBoolean            isResizable;
ODBoolean            isFloating;
ODBoolean            shouldSave;
ODBoolean            shouldDispose;
ODPart              *rootPart;
ODTypeToken          viewType;
ODTypeToken          presentation;
ODFrame              *sourceFrame;
ODWindow             *rv;

rv = RegisterWindow(newWindow, frameType,
    isRootWindow, isResizable, isFloating,
    shouldSave, shouldDispose, rootPart,
    viewType, presentation, sourceFrame);
```

RegisterWindow Parameter - newWindow

newWindow (ODPlatformWindow) - input
A platform-specific window structure.

RegisterWindow Parameter - frameType

frameType (ODType) - input
The type of root frame for the window object. This parameter must be set to one of the following values:

- kODFrameObject
A regular frame.
 - kODNonPersistentFrameObject
A nonpersistent frame object.
-

RegisterWindow Parameter - isRootWindow

isRootWindow (ODBoolean) - input
A flag indicating whether the window object is to be a root window.

- kODTrue
The window object is to be a root window.
 - kODFalse
The window object is to be a regular window.
-

RegisterWindow Parameter - isResizable

isResizable (ODBoolean) - input
A flag indicating whether the window object is to be resizable.

- kODTrue
The window object is to be resizable.
 - kODFalse
The window object is not to be resizable.
-

RegisterWindow Parameter - isFloating

isFloating (ODBoolean) - input
A flag indicating whether the window object is to be a floating window.

- kODTrue

kODFalse The window object is to be a floating window.
The window object is not to be a floating window.

RegisterWindow Parameter - shouldSave

shouldSave (ODBoolean) - input
A flag indicating whether the window object is to be saved in its draft.

kODTrue The window object is to be saved in its draft.
kODFalse The window object is not to be saved in its draft.

RegisterWindow Parameter - shouldDispose

shouldDispose (ODBoolean) - input
A flag indicating whether the platform window should be disposed of when this window object is deleted.

kODTrue The window object should be disposed of.
kODFalse The window object should not be disposed of.

RegisterWindow Parameter - rootPart

rootPart (ODPart *) - input
A reference to the part associated with the root frame of the window.

RegisterWindow Parameter - viewType

viewType (ODTypeToken) - input
A tokenized string representing the view type for the root frame of the window.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame Framed view type.
kODViewAsLargeIcon Large-icon (standard) view type
kODViewAsSmallIcon Small-icon view type.
kODViewAsThumbNail Thumbnail view type

RegisterWindow Parameter - presentation

presentation (ODTypeToken) - input

A tokenized string representing the presentation type for the root frame of the window.

RegisterWindow Parameter - sourceFrame

sourceFrame (ODFrame *) - input

A reference to the frame from which the window object was opened (used when an embedded frame is opened into a window) or kODNULL if the frame does not exist.

RegisterWindow Return Value - rv

rv (ODWindow *) - returns

A reference to the newly created window object.

RegisterWindow - Parameters

newWindow (ODPlatformWindow) - input

A platform-specific window structure.

frameType (ODType) - input

The type of root frame for the window object. This parameter must be set to one of the following values:

kODFrameObject

A regular frame.

kODNonPersistentFrameObject

A nonpersistent frame object.

isRootWindow (ODBoolean) - input

A flag indicating whether the window object is to be a root window.

kODTrue

The window object is to be a root window.

kODFalse

The window object is to be a regular window.

isResizable (ODBoolean) - input

A flag indicating whether the window object is to be resizable.

kODTrue

The window object is to be resizable.

kODFalse

The window object is not to be resizable.

isFloating (ODBoolean) - input

A flag indicating whether the window object is to be a floating window.

kODTrue The window object is to be a floating window.
kODFalse The window object is not to be a floating window.

shouldSave ([ODBoolean](#)) - input
A flag indicating whether the window object is to be saved in its draft.

kODTrue The window object is to be saved in its draft.
kODFalse The window object is not to be saved in its draft.

shouldDispose ([ODBoolean](#)) - input
A flag indicating whether the platform window should be disposed of when this window object is deleted.

kODTrue The window object should be disposed of.
kODFalse The window object should not be disposed of.

rootPart (ODPart *) - input
A reference to the part associated with the root frame of the window.

viewType ([ODTypeToken](#)) - input
A tokenized string representing the view type for the root frame of the window.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame Framed view type.
kODViewAsLargeIcon Large-icon (standard) view type
kODViewAsSmallIcon Small-icon view type.
kODViewAsThumbNail Thumbnail view type

presentation ([ODTypeToken](#)) - input
A tokenized string representing the presentation type for the root frame of the window.

sourceFrame (ODFrame *) - input
A reference to the frame from which the window object was opened (used when an embedded frame is opened into a window) or kODNULL if the frame does not exist.

rv (ODWindow *) - returns
A reference to the newly created window object.

RegisterWindow - Remarks

Your part calls this method to create a window object when there is no pre-existing root frame. You should create the platform-specific window as invisible. After calling this method, you should call the [Show](#) method of the returned window to make it visible.

This method initializes the reference count of the returned window. When you have finished using that window, you should call its [Release](#) method.

RegisterWindow - Exception Handling

kODErrCannotCreateWindow There is not enough memory to create the new window object.

RegisterWindow - Related Methods

Related Methods

- [ODSession::Tokenize](#)
- [ODWindow::Show](#)
- [ODWindowState::RegisterWindowForFrame](#)

RegisterWindow - Topics

Class:

[ODWindowState](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Exception Handling](#)
[Related Methods](#)

RegisterWindowForFrame

RegisterWindowForFrame - Syntax

This method creates a window object for the specified platform-specific window and root frame.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODPlatformWindow    newWindow;
ODFrame             *frame;
ODBoolean           isRootWindow;
ODBoolean           isResizable;
ODBoolean           isFloating;
ODBoolean           shouldSave;
ODBoolean           shouldDispose;
ODFrame             *sourceFrame;
ODWindow            *rv;

rv = RegisterWindowForFrame(newWindow, frame,
    isRootWindow, isResizable, isFloating,
    shouldSave, shouldDispose, sourceFrame);
```

RegisterWindowForFrame Parameter - newWindow

newWindow (ODPlatformWindow) - input
A platform-specific window.

RegisterWindowForFrame Parameter - frame

frame (ODFrame *) - input
A reference to the root frame of the window object.

RegisterWindowForFrame Parameter - isRootWindow

isRootWindow (ODBoolean) - input
A flag indicating whether the window object is to be a root window.

kODTrue	The window object is to be a root window.
kODFalse	The window object is to be a regular window.

RegisterWindowForFrame Parameter - isResizable

isResizable (ODBoolean) - input
A flag indicating whether the window object is to be resizable.

kODTrue	The window object is to be resizable.
kODFalse	The window object is not to be resizable.

RegisterWindowForFrame Parameter - isFloating

isFloating (ODBoolean) - input
A flag indicating whether the window object is to be a floating window.

kODTrue	The window object is to be a floating window.
kODFalse	The window object is not to be a floating window.

RegisterWindowForFrame Parameter - shouldSave

shouldSave (ODBoolean) - input

A flag indicating whether the window object is to be saved in its draft.

kODTrue

The window object is to be saved in its draft.

kODFalse

The window object is not to be saved in its draft.

RegisterWindowForFrame Parameter - shouldDispose

shouldDispose (ODBoolean) - input

A flag indicating whether the platform window should be disposed of when this window object is deleted.

kODTrue

The window object should be disposed of.

kODFalse

The window object should not be disposed of.

RegisterWindowForFrame Parameter - sourceFrame

sourceFrame (ODFrame *) - input

A reference to the frame from which the window object was opened (used when an embedded frame is opened into a window) or kODNULL if the frame does not exist.

RegisterWindowForFrame Return Value - rv

rv (ODWindow *) - returns

A reference to the newly created window.

RegisterWindowForFrame - Parameters

newWindow (ODPlatformWindow) - input

A platform-specific window.

frame (ODFrame *) - input

A reference to the root frame of the window object.

isRootWindow (ODBoolean) - input

A flag indicating whether the window object is to be a root window.

kODTrue

The window object is to be a root window.

kODFalse

The window object is to be a regular window.

isResizable ([ODBoolean](#)) - input

A flag indicating whether the window object is to be resizable.

kODTrue

The window object is to be resizable.

kODFalse

The window object is not to be resizable.

isFloating ([ODBoolean](#)) - input

A flag indicating whether the window object is to be a floating window.

kODTrue

The window object is to be a floating window.

kODFalse

The window object is not to be a floating window.

shouldSave ([ODBoolean](#)) - input

A flag indicating whether the window object is to be saved in its draft.

kODTrue

The window object is to be saved in its draft.

kODFalse

The window object is not to be saved in its draft.

shouldDispose ([ODBoolean](#)) - input

A flag indicating whether the platform window should be disposed of when this window object is deleted.

kODTrue

The window object should be disposed of.

kODFalse

The window object should not be disposed of.

sourceFrame ([ODFrame *](#)) - input

A reference to the frame from which the window object was opened (used when an embedded frame is opened into a window) or kODNULL if the frame does not exist.

rv ([ODWindow *](#)) - returns

A reference to the newly created window.

RegisterWindowForFrame - Remarks

Your part calls this method to recreate the windows of an existing document. You should create the platform-specific window as invisible. After calling this method, you should call the [Show](#) method of the returned window to make the window visible.

This window initializes the reference count of the returned window object. When you have finished using that window object, you should call its [Acquire](#) method.

RegisterWindowForFrame - Exception Handling

kODErrCannotCreateWindow

There is not enough memory to create the new window object.

RegisterWindowForFrame - Related Methods

Related Methods

- [ODWindow::Show](#)
- [ODWindowState::RegisterWindow](#)

RegisterWindowForFrame - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

SetBaseMenuBar

SetBaseMenuBar - Syntax

This method is called by the document shell or container application to install the specified menu-bar object as the base menu for parts to copy.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>

ODMenuBar      *theMenuBar;

SetBaseMenuBar(theMenuBar);
```

SetBaseMenuBar Parameter - theMenuBar

theMenuBar (ODMenuBar *) - input

A reference to the menu-bar object to be installed as the base menu bar.

SetBaseMenuBar - Return Value

None.

SetBaseMenuBar - Parameters

theMenuBar (ODMenuBar *) - input

A reference to the menu-bar object to be installed as the base menu bar.

None.

SetBaseMenuBar - Remarks

On OS/2, the base menu bar contains the **Document**, **Edit**, **View**, and **Help** menus.

SetBaseMenuBar - Topics

Class:

ODWindowState

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

SetBasePopup (OS/2)

SetBasePopup (OS/2) - Syntax

This method is called by the document shell or container application to install the pop-up menu object used by all parts.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>
```

```
ODPopup      *thePopup;
```

```
SetBasePopup(thePopup);
```

SetBasePopup (OS/2) Parameter - thePopup

thePopup (ODPopup *) - input
A reference to the pop-up menu object.

SetBasePopup (OS/2) - Return Value

None.

SetBasePopup (OS/2) - Parameters

thePopup (ODPopup *) - input
A reference to the pop-up menu object.

None.

SetBasePopup (OS/2) - Topics

Class:
ODWindowState

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

SetDefaultWindowTitles

SetDefaultWindowTitles - Syntax

This method is called by the document shell or container application to synchronize window titles with the file name of the document of the specified draft.

```
#define INCL_ODWINDOWSTATE
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraft    *draft;
```

```
SetDefaultWindowTitles(draft);
```

SetDefaultWindowTitles Parameter - draft

draft (ODDraft *) - input
A reference to the open draft object.

SetDefaultWindowTitles - Return Value

None.

SetDefaultWindowTitles - Parameters

draft (ODDraft *) - input
A reference to the open draft object.

None.

SetDefaultWindowTitles - Topics

Class:
ODWindowState

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

RegistryManager

Class Definition File: RMANAGER.IDL

Class Hierarchy
SOMObject
 RegistryManager

Description

The Registry Manager provides access to the [ODPartHandlerRegistry](#) and the Component Manager. It also sets up communication with the

data base server.

Methods

The methods defined by the ODRegistryManager class include:

- [GetComponentManager](#)
- [GetODPartHandlerRegistry](#)

Overridden Methods

There are currently no methods overridden by the ODRegistryManager class.

GetComponentManager (OS/2)

GetComponentManager (OS/2) - Syntax

This method gain access to the component registries.

```
#define INCL_ODCOMPONENT
#define INCL_ODREGISTRYMANAGER
#define INCL_ODAPI
#include <os2.h>
```

```
ComponentManager      *rv;

rv = GetComponentManager();
```

GetComponentManager (OS/2) Return Value - rv

rv (ComponentManager *) - returns

A pointer to the component registries. When it is finished using this value, the client code is responsible for freeing it using the somFree method.

GetComponentManager (OS/2) - Parameters

rv (ComponentManager *) - returns

A pointer to the component registries. When it is finished using this value, the client code is responsible for freeing it using the somFree method.

GetComponentManager (OS/2) - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetODPartHandlerRegistry (OS/2)

GetODPartHandlerRegistry (OS/2) - Syntax

This method queries and updates part-handler information.

```
#define INCL_ODCOMPONENT
#define INCL_ODREGISTRYMANAGER
#define INCL_ODAPI
#include <os2.h>

ODPartHandlerRegistry    *rv;

rv = GetODPartHandlerRegistry();
```

GetODPartHandlerRegistry (OS/2) Return Value - rv

rv (ODPartHandlerRegistry *) - returns

A pointer to the part-handler registry. When it is finished using this value, the client code is responsible for freeing it using the somFree method.

GetODPartHandlerRegistry (OS/2) - Parameters

rv (ODPartHandlerRegistry *) - returns

A pointer to the part-handler registry. When it is finished using this value, the client code is responsible for freeing it using the somFree method.

GetODPartHandlerRegistry (OS/2) - Topics

Class:

RegistryManager

Select an item:

[Syntax](#)

OpenDoc Metaclasses and Methods

This chapter contains an alphabetic listing of the OpenDoc object metaclasses and their methods. These sections contain technical reference information. See the *OpenDoc Programming Guide* for OpenDoc guide information. For information on the System Object Model (SOM), see the *System Object Model Guide and Reference*.

M_ODFacet

Class Definition File: ODFACET.IDL

Class Hierarchy

Description

Methods

The methods defined by the M_ODFacet metaclass include:

- [clsGetFacetFromHWND](#)

Overridden Methods

There are currently no methods overridden by the M_ODFacet metaclass.

clsGetFacetFromHWND (OS/2)

clsGetFacetFromHWND (OS/2) - Syntax

This class method returns a reference to the facet corresponding to the specified window handle.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

HWND      hwnd;
ODFacet   *rv;

rv = clsGetFacetFromHWND(hwnd);
```

clsGetFacetFromHWND (OS/2) Parameter - hwnd

hwnd ([HWND](#)) - input
The window handle whose corresponding facet is to be returned.

clsGetFacetFromHWND (OS/2) Return Value - rv

rv (ODFacet *) - returns
A pointer to the corresponding facet. A return value of KODNULL indicates that there is no corresponding facet.

clsGetFacetFromHWND (OS/2) - Parameters

hwnd ([HWND](#)) - input
The window handle whose corresponding facet is to be returned.

rv (ODFacet *) - returns
A pointer to the corresponding facet. A return value of KODNULL indicates that there is no corresponding facet.

clsGetFacetFromHWND (OS/2) - Remarks

This class method provides the inverse function of the facet's [GetFacetHWND](#) method.

clsGetFacetFromHWND (OS/2) - Topics

Class:
ODFacet

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

M_ODPart

Class Definition File: PART.IDL

Class Hierarchy

Description

The M_ODPart handler metaclass is provided to part developers. Part developers are required to override the five class methods, which provide registration information.

Methods

The methods defined by the M_ODPart class include:

- [clsGetOLE2ClassId](#)
- [clsGetODPartHandlerDisplayName](#)
- [clsGetODPartHandlerName](#)
- [clsGetODPartKinds](#)
- [clsGetWindowsIconFileName](#)

Overridden Methods

There are currently no methods overridden by the M_ODPart class.

clsGetOLE2ClassId (OS/2)

clsGetOLE2ClassId (OS/2) - Syntax

This method returns the OLE2 class ID expressed as a string.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = clsGetOLE2ClassId();
```

clsGetOLE2ClassId (OS/2) Return Value - rv

rv ([string](#)) - returns

The OLE2 class ID expressed as a string. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetOLE2ClassId (OS/2) - Parameters

rv ([string](#)) - returns

The OLE2 class ID expressed as a string. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetOLE2ClassId (OS/2) - Remarks

This ID will be used to identify this part to OLE containers that want to embed this part. This is required only if interoperability with OLE2 is desired. This method can return a null string.

clsGetOLE2ClassId (OS/2) - Topics

Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

clsGetODPartHandlerDisplayName (OS/2)

clsGetODPartHandlerDisplayName (OS/2) - Syntax

This method returns the part handler's display name.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = clsGetODPartHandlerDisplayName();
```

clsGetODPartHandlerDisplayName (OS/2) Return Value - rv

rv ([string](#)) - returns

A string containing the part handler's display name. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetODPartHandlerDisplayName (OS/2) - Parameters

rv ([string](#)) - returns

A string containing the part handler's display name. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetODPartHandlerDisplayName (OS/2) - Topics

Class:

ODPart

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)

clsGetODPartHandlerName (OS/2)

clsGetODPartHandlerName (OS/2) - Syntax

This method returns the part handler's name.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ISOString    rv;
```

```
rv = clsGetODPartHandlerName();
```

clsGetODPartHandlerName (OS/2) Return Value - rv

rv ([ISOString](#)) - returns

The part handler name. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetODPartHandlerName (OS/2) - Parameters

rv ([ISOString](#)) - returns

The part handler name. When it is finished using this value, the client code is responsible for freeing the value by calling somFree.

clsGetODPartHandlerName (OS/2) - Topics

Class:

ODPart

Select an item:

[Syntax](#)

clsGetODPartKinds (OS/2)

clsGetODPartKinds (OS/2) - Syntax

This method returns a sequence containing all of the part kinds supported by this part handler.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

_IDL_SEQUENCE_PartKindInfo    rv;

rv = clsGetODPartKinds();
```

clsGetODPartKinds (OS/2) Return Value - rv

rv ([_IDL_SEQUENCE_PartKindInfo](#)) - returns

A sequence containing all the part kinds supported by this part handler. When it is finished using the sequence, the client code is responsible for freeing each of the objects in the sequence and the sequence buffer by calling somFree.

clsGetODPartKinds (OS/2) - Parameters

rv ([_IDL_SEQUENCE_PartKindInfo](#)) - returns

A sequence containing all the part kinds supported by this part handler. When it is finished using the sequence, the client code is responsible for freeing each of the objects in the sequence and the sequence buffer by calling somFree.

clsGetODPartKinds (OS/2) - Topics

Class:

ODPart

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

clsGetWindowsIconFileName (OS/2)

clsGetWindowsIconFileName (OS/2) - Syntax

This method returns the file name of the icon identifying this part to OLE containers.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = clsGetWindowsIconFileName();
```

clsGetWindowsIconFileName (OS/2) Return Value - rv

rv ([string](#)) - returns
The name of a window icon file.

clsGetWindowsIconFileName (OS/2) - Parameters

rv ([string](#)) - returns
The name of a window icon file.

clsGetWindowsIconFileName (OS/2) - Remarks

This icon will be used to identify this part to OLE containers that want to embed this part. The method can return a null string. This is required only if interoperability with OLE2 is desired. When finished using the value returned by this method, the client code is responsible for freeing the string by calling SOMFree.

clsGetWindowsIconFileName (OS/2) - Topics

Class:
ODPart

Select an item:
[Syntax](#)

Workplace Shell Classes and Methods

This section contains an alphabetic list of the Workplace Shell methods that are available to an application for using and controlling OpenDoc objects. See *Workplace Shell Programming Reference* for documentation of Workplace Shell methods.

ODwps

Class definition file: ODWPS.IDL

Class hierarchy

```
SOMObject
  WPObject
    WPFileSystem
      WPDataFile
        ODwps
```

Description

This class allows OpenDoc documents to operate as Workplace Shell objects. These actions include double-clicking to open a document.

For OpenDoc class objects not created from the Workplace Shell, the Workplace Shell needs a way to recognize the file as an OpenDoc file. The following two methods can be used:

- If a file has an extended attribute type of "OpenDoc Document", the Workplace Shell recognizes it as an OpenDoc file and creates an object of OpenDoc class when needed.
- A file with an extension of .OD is recognized as an OpenDoc file.

New methods

There are currently no methods defined by the ODwps class.

Overridden methods

The following lists show the methods overridden by the ODwps class.

WPObject methods

The following list shows the methods that override the corresponding method from the WPObject class. These methods are overridden in order to modify the behavior defined by the ancestor class.

- wpAddSettingsPages
- wpFormatDragItem
- wpInitData
- wpMenuItemSelected
- wpModifyPopupMenu
- wpOpen
- wpPrint
- wpQueryDefaultView
- wpSetupOnce
- wpUnInitData
- wpclsCreateDefaultTemplates
- wpclsInitData
- wpclsQueryDefaultHelp
- wpclsQueryDefaultView
- wpclsQueryIconData
- wpclsQueryStyle
- wpclsQueryTitle

WPFileSystem methods

The following list shows the methods that override the corresponding method from the WPFileSystem class. These methods are overridden in order to modify the behavior defined by the ancestor class.

- wpclsQueryInstanceFilter
- wpclsQueryInstanceType

WPDataFile methods

The following list shows the methods that override the corresponding method from the WPDataFile class. These methods are overridden in order to modify the behavior defined by the ancestor class.

- wpAddFileTypePage

wpAddFileTypePage

wpAddFileTypePage - Syntax

This instance method is called to remove the *Type* page from [ODwps](#) objects Settings Notebook.

This method is an override of the corresponding method from the WPDataFile ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;      /* Pointer to the object on which the method is being invoked. */
HWND       hwndNotebook; /* Settings notebook handle. */
ULONG      rc;            /* Page identifier. */

rc = wpAddFileTypePage(somSelf, hwndNotebook);
```

wpAddFileTypePage Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpAddFileTypePage Parameter - hwndNotebook

hwndNotebook ([HWND](#)) - input

Settings notebook handle.

wpAddFileTypePage Return Value - rc

rc ([ULONG](#)) - returns
Page identifier.

0	Error occurred
PageId	Identifier for the inserted page.

wpAddFileTypePage - Parameters

somSelf ([ODwps *](#)) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

hwndNotebook ([HWND](#)) - input
Settings notebook handle.

rc ([ULONG](#)) - returns
Page identifier.

0	Error occurred
PageId	Identifier for the inserted page.

wpAddFileTypePage - Remarks

The TYPE of [ODwps](#) files is "OpenDoc Document" and cannot be changed by the end user.

wpAddFileTypePage - Usage

This method must be called only from within an override of the `wpAddSettingsPages` method.

wpAddFileTypePage - Topics

Class:
[ODwps](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[Usage](#)

wpFormatDragItem

wpFormatDragItem - Syntax

This instance method is called to allow the object to format its drag information when the user starts to drag it.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;      /* Pointer to the object on which the method is being invoked. */
PDRAGITEM  pDragItem;     /* Address of the drag item. */
BOOL       rc;            /* Success indicator. */

rc = wpFormatDragItem(somSelf, pDragItem);
```

wpFormatDragItem Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpFormatDragItem Parameter - pDragItem

pDragItem ([PDRAGITEM](#)) - input

Address of the drag item.

This method adds the DRF_OPENDOCDOCUMENT constant to the valid formats for the *hstrRMF* field of the [DRAGITEM](#) structure.

wpFormatDragItem Return Value - rc

rc ([BOOL](#)) - returns

Success indicator.

TRUE

FALSE

Successful completion.

Error occurred.

wpFormatDragItem - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

pDragItem ([PDRAGITEM](#)) - input
Address of the drag item.

This method adds the DRF_OPENDOCDOCUMENT constant to the valid formats for the *hstrRMF* field of the [DRAGITEM](#) structure.

rc ([BOOL](#)) - returns
Success indicator.

TRUE	Successful completion.
FALSE	Error occurred.

wpFormatDragItem - Remarks

This method enables the direct manipulation of this object by initializing the [DRAGITEM](#) structure.

wpFormatDragItem - Usage

This method is generally called only by the system when the user first starts to drag the object.

wpFormatDragItem - How to Override

This method is generally overridden by classes that require special processing to initiate a drag or drop operation.

wpFormatDragItem - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[How to Override](#)
[Usage](#)

wpInitData

wpInitData - Syntax

This instance method is called to create an object window for notification purposes.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the object on which the method is being invoked. */

wpInitData(somSelf);
```

wpInitData Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpInitData - Return Value

There is no return value for this method.

wpInitData - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

There is no return value for this method.

wpInitData - Remarks

This routine is called by the system when the object is created or when it is awakened from the dormant state so that it can initialize all of its instance variables to a known state. By default, memory allocated to instance variables is zero-filled. Note that this method is called before the object's state is known, so it is very important that the object does not try to process any other method while processing this method. Should an object need extra initialization that requires it to invoke other methods, this should be done from [wpRestoreState](#).

When the object is first created, [wpSetupOnce](#) should be overridden to perform initialization that is required only once.

The parent method must be called before any processing is done by your overriding method.

wpInitData - Usage

This method is generally called only by the system when the object is awakened.

wpInitData - How to Override

Subclasses that override this method should call the parent method before any special processing.

wpInitData - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

wpMenuItemSelected

wpMenuItemSelected - Syntax

This instance method processes an OpenDoc specific pop-up menu selection.

This method is an override of the corresponding method from the WPObj ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>
```

```
ODwps      *somSelf;    /* Pointer to the object on which the method is being invoked. */
HWND       hwndFrame;   /* Handle to the frame window. */
ULONG      ulMenuId;    /* ID of the selected pop-up menu. */
```

```
wpMenuItemSelected(somSelf, hwndFrame, ulMenuId);
```

wpMenuItemSelected Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpMenuItemSelected Parameter - hwndFrame

hwndFrame ([HWND](#)) - input
Handle to the frame window.

wpMenuItemSelected Parameter - ulMenuId

ulMenuId ([ULONG](#)) - input
ID of the selected pop-up menu.

For a listing of WPMENUIDs, see the individual object classes.

wpMenuItemSelected - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

hwndFrame ([HWND](#)) - input
Handle to the frame window.

ulMenuId ([ULONG](#)) - input
ID of the selected pop-up menu.

For a listing of WPMENUIDs, see the individual object classes.

wpMenuItemSelected - Remarks

Class-specific menu IDs should be above WPMENUID_USER.

wpMenuItemSelected - Usage

This method is generally called only by the system when a pop-up menu item is selected.

wpMenuItemSelected - How to Override

This method should be overridden to process class-specific menu-item actions or to modify the behavior of a menu-item action provided by an ancestor class.

wpMenuItemSelected - Related Methods

Related Methods

- [wpFilterPopupMenu](#)
 - [wpInsertPopupMenuItems](#)
 - [wpMenuItemHelpSelected](#)
 - [wpMenuItemSelected](#)
 - [wpModifyPopupMenu](#)
-

wpMenuItemSelected - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

wpModifyPopupMenu

wpModifyPopupMenu - Syntax

This overridden instance method adds the OpenDoc views available for this object to the "Open" pop-up menu.

This method is an override of the corresponding method from the WPObjec ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;      /* Pointer to the object on which the method is being invoked. */
HWND       hwndMenu;      /* Menu handle. */
HWND       hwndCnr;       /* Handle to container control window. */
ULONG      ulPosition;    /* Position to insert menu items. */
BOOL       rc;            /* Success indicator. */

rc = wpModifyPopupMenu(somSelf, hwndMenu,
                      hwndCnr, ulPosition);
```

wpModifyPopupMenu Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.
Points to an object of class [ODwps](#).

wpModifyPopupMenu Parameter - hwndMenu

hwndMenu ([HWND](#)) - input
Menu handle.

wpModifyPopupMenu Parameter - hwndCnr

hwndCnr ([HWND](#)) - input
Handle to container control window.

wpModifyPopupMenu Parameter - ulPosition

ulPosition ([ULONG](#)) - input
Position to insert menu items.

wpModifyPopupMenu Return Value - rc

rc ([BOOL](#)) - returns
Success indicator.

TRUE	Successful completion.
FALSE	Error occurred.

wpModifyPopupMenu - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

hwndMenu ([HWND](#)) - input
Menu handle.

hwndCnr ([HWND](#)) - input
Handle to container control window.

ulPosition ([ULONG](#)) - input
Position to insert menu items.

rc ([BOOL](#)) - returns
Success indicator.

TRUE
FALSE

Successful completion.
Error occurred.

wpModifyPopupMenu - Remarks

Class-specific menu IDs should be above WPMENUID_USER. This method should be called only if the current pop-up menu applies to objects of the same class.

wpModifyPopupMenu - Usage

This method is generally called by the system when a request to display the object's pop-up menu is made. This method is called following a call to the wpFilterPopupMenu method.

wpModifyPopupMenu - How to Override

This method should be overridden in order to add class-specific actions to the object's pop-up menu. Descendant classes can remove these actions by processing the wpFilterPopupMenu method.

wpModifyPopupMenu - Related Methods

Related Methods

- wpFilterPopupMenu
- wpInsertPopupMenuItems
- wpMenuItemHelpSelected
- [wpMenuItemSelected](#)
- wpModifyPopupMenu

wpModifyPopupMenu - Topics

Class:
ODwps

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[How to Override](#)
[Usage](#)
[Related Methods](#)

wpOpen

wpOpen - Syntax

This method invokes the DocShell program to open a document.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the object on which the method is being invoked. */
HWND       hwndCnr; /* Handle of the container window. */
ULONG      ulView; /* Specifies which view to open. */
ULONG      ulparam; /* Open view parameter. */
HWND       rc; /* Success indicator. */

rc = wpOpen(somSelf, hwndCnr, ulView, ulparam);
```

wpOpen Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpOpen Parameter - hwndCnr

hwndCnr (HWND) - input

Handle of the container window.

Handle of the container window from which the object is opened. This value may be set to NULLHANDLE.

wpOpen Parameter - ulView

ulView ([ULONG](#)) - input
Specifies which view to open.

OPEN_CONTENTS	Open contents view.
OPEN_DEFAULT	Open default view (same as double-click).
OPEN_DETAILS	Open details view.
OPEN_HELP	Display HelpPanel.
OPEN_RUNNING	Execute object.
OPEN_SETTINGS	Open Settings notebook.
OPEN_TREE	Open tree view.
OPEN_USER	Class-specific views have a greater value than this.

wpOpen Parameter - ulparam

ulparam ([ULONG](#)) - input
Open view parameter.

This value is (reserved = NULL) for views supported by the WPObjct class.

wpOpen Return Value - rc

rc ([HWND](#)) - returns
Success indicator.

NULLHANDLE	Error occurred.
Other	Handle to either window created or program executed.

wpOpen - Parameters

somSelf ([ODwps *](#)) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

hwndCnr ([HWND](#)) - input
Handle of the container window.

Handle of the container window from which the object is opened. This value may be set to NULLHANDLE.

ulView ([ULONG](#)) - input
Specifies which view to open.

OPEN_CONTENTS	Open contents view.
---------------	---------------------

OPEN_DEFAULT	Open default view (same as double-click).
OPEN_DETAILS	Open details view.
OPEN_HELP	Display HelpPanel.
OPEN_RUNNING	Execute object.
OPEN_SETTINGS	Open Settings notebook.
OPEN_TREE	Open tree view.
OPEN_USER	Class-specific views have a greater value than this.

ulparam ([ULONG](#)) - input
Open view parameter.

This value is (reserved = NULL) for views supported by the WPObj class.

rc ([HWND](#)) - returns
Success indicator.

NULLHANDLE	Error occurred.
Other	Handle to either window created or program executed.

wpOpen - Remarks

This method invokes DOCSHELL.EXE against the name of the file containing the object.

wpOpen - Usage

This method can be called at any time in order to open a view of an object.

wpOpen - How to Override

This method should be overridden in order to process class-specific open views. This method can also be overridden in order to modify the behavior defined by an ancestor class. When wpOpen is overridden to implement a user-defined view, it should call wpAddToObjUseList and wpRegisterView to ensure that the new view is registered by the Workplace Shell as a view of the object.

wpOpen - Related Methods

Related Methods

- [wpClose](#)

wpOpen - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

wpSetupOnce

wpSetupOnce - Syntax

This instance method is called to allow the newly created object to initialize itself.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;          /* Pointer to the object on which the method is being invoked. */
PSZ        pszSetupString;    /* Class specific setup parameters for an object. */
BOOL       rc;               /* Success indicator. */

rc = wpSetupOnce(somSelf, pszSetupString);
```

wpSetupOnce Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpSetupOnce Parameter - pszSetupString

pszSetupString ([PSZ](#)) - input

Class specific setup parameters for an object.

wpSetupOnce Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE

Successful completion.

FALSE

Error occurred.

wpSetupOnce - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class **ODwps**.

pszSetupString (PSZ) - input
Class specific setup parameters for an object.

rc (BOOL) - returns
Success indicator.

TRUE

Successful completion.

FALSE

Error occurred.

wpSetupOnce - Remarks

If wpSetupOnce returns FALSE, the creation of the object is terminated.

The *pszSetupString* contains a series of "keyname=value" pairs that change the behavior of the object. Multiple "keyname=value" pairs are separated by semicolons. For example:

```
"key=value;key2=value1,value2;"
```

If you want a literal semicolon inside one of your fields you must type the following:

```
^;          A literal semicolon.
```

Each object class documents the keynames and the parameters it expects to see immediately following. Note that all parameters have safe defaults, so it is never required to pass parameters to an object.

For a listing of setup strings, see the individual object classes. This method adds the following setup string:

KEYNAME	VALUE	DESCRIPTION
SEEDER	fully-qualified file name	This allows an object to be created (for example, using the WinCreateObject function) with initial content. The content of the specified file name are written to the new object to "seed" it. NOTE: This only has an effect when an ODwps class object is initially created, and only when it

```
has no initial content;  
that is, if an ODwps object  
is created by copying an  
existing one, this setup  
string has no effect.
```

wpSetupOnce - Usage

This method is generally only called by the system during the processing of wpclsNew and WinCreateObject.

For more information about WinCreateObject, see the *Presentation Manager Programming Reference* .

wpSetupOnce - How to Override

Subclasses that override this method should call the parent before any special processing.

wpSetupOnce - Related Methods

Related Methods

- [wpSetup](#)
-

wpSetupOnce - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[How to Override](#)
[Usage](#)
[Related Methods](#)

wpUnInitData

wpUnInitData - Syntax

This instance method is called to allow the object to free allocated resources.

This method is an override of the corresponding method from the WPObjec ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the object on which the method is being invoked. */

wpUnInitData(somSelf);
```

wpUnInitData Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

wpUnInitData - Return Value

There is no return value for this method.

wpUnInitData - Parameters

somSelf (ODwps *) - input
Pointer to the object on which the method is being invoked.

Points to an object of class [ODwps](#).

There is no return value for this method.

wpUnInitData - Remarks

Subclasses that override this method must call the parent method after any special processing.

wpUnInitData - Usage

This method is generally called only by the system when the object is made dormant. The object is made dormant when it is destroyed or when there are no open views and the folder containing the object are not open.

wpUnInitData - How to Override

This method is overridden to de-allocate resources allocated during the processing of wpInitData.

The parent method must be called after you have completed your own processing.

wpUnInitData - Related Methods

Related Methods

- [wpInitData](#)
-

wpUnInitData - Topics

Class:

ODwps

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[How to Override](#)
[Usage](#)
[Related Methods](#)

wpclsCreateDefaultTemplates

wpclsCreateDefaultTemplates - Syntax

This method is called to allow the specified class to create default template instances of its class.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */
WPFolder   *Folder;  /* Pointer to the folder in which to create the templates. */
BOOL       rv;        /* Flag indicating whether the class creates the templates. */

rv = wpclsCreateDefaultTemplates(somSelf,
    Folder);
```

wpclsCreateDefaultTemplates Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

wpclsCreateDefaultTemplates Parameter - Folder

Folder (WPFolder *) - input
Pointer to the folder in which to create the templates.

Points to an object of class wpFolder.

wpclsCreateDefaultTemplates Return Value - rv

rv ([BOOL](#)) - returns
Flag indicating whether the class creates the templates.

This method does not create any default templates in the OS/2 template folder; therefore, this flag returns FALSE.

Templates can be made of ODwps objects that contain parts.

wpclsCreateDefaultTemplates - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

Folder (WPFolder *) - input
Pointer to the folder in which to create the templates.

Points to an object of class wpFolder.

rv ([BOOL](#)) - returns
Flag indicating whether the class creates the templates.

This method does not create any default templates in the OS/2 template folder; therefore, this flag returns FALSE.

Templates can be made of ODwps objects that contain parts.

wpclsCreateDefaultTemplates - Usage

This method is generally called only by the system when the class is registered. A class is registered by a call to the WinRegisterObjectClass function.

When the system calls this method, "Folder" is a pointer to the Templates folder.

wpclsCreateDefaultTemplates - How to Override

This method should be overridden by classes that need to create default template instances of their class.

wpclsCreateDefaultTemplates - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[How to Override](#)

[Usage](#)

wpclsInitData

wpclsInitData - Syntax

This method is called to allow the class object to initialize its instance data.

This method is an override of the corresponding method from the WPObjcet ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */

wpclsInitData(somSelf);
```

wpclsInitData Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the [ODwps](#) class object.

wpclsInitData - Return Value

There is no return value for this method.

wpclsInitData - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

There is no return value for this method.

wpclsInitData - Remarks

This method is called immediately after the class object is first awakened, and then sets the type and title strings from the MRI resource.

wpclsInitData - Usage

This method is generally called only by the system when the class object is awakened. The class object is awakened when the first instance of this class is either awakened or newly created. It is made dormant again when the last instance of this class is made dormant.

wpclsInitData - How to Override

Any class that has metaclass instance variables should override this method so that those variables are all initially in a known state. It is essential to pass this method onto the parent class object before performing any override processing.

wpclsInitData - Topics

Class:
[ODwps](#)

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)
[How to Override](#)
[Usage](#)

wpclsQueryDefaultHelp

wpclsQueryDefaultHelp - Syntax

This method is called to allow the class object to specify its default help panel for its instances.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;      /* Pointer to the ODwps class object. */
PULONG      HelpPanelId;  /* Pointer to the help panel which provides help for this class. */
PSZ         HelpLibrary;  /* Pointer to the buffer in which to place the name of the Help library. */
ULONG       rc;           /* Success indicator. */

rc = wpclsQueryDefaultHelp(somSelf, HelpPanelId,
                          HelpLibrary);
```

wpclsQueryDefaultHelp Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the ODwps class object.

wpclsQueryDefaultHelp Parameter - HelpPanelId

HelpPanelId (PULONG) - output
Pointer to the help panel which provides help for this class.

wpclsQueryDefaultHelp Parameter - HelpLibrary

HelpLibrary (PSZ) - output
Pointer to the buffer in which to place the name of the Help library.

This buffer should be at least the length of CCHMAXPATH bytes.

wpclsQueryDefaultHelp Return Value - rc

rc (ULONG) - returns

Success indicator.

TRUE
FALSE

Successful completion.
Error occurred.

wpclsQueryDefaultHelp - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

HelpPanelId ([PULONG](#)) - output
Pointer to the help panel which provides help for this class.

HelpLibrary ([PSZ](#)) - output
Pointer to the buffer in which to place the name of the Help library.

This buffer should be at least the length of CCHMAXPATH bytes.

rc ([ULONG](#)) - returns
Success indicator.

TRUE
FALSE

Successful completion.
Error occurred.

wpclsQueryDefaultHelp - Remarks

This class method is called during the default processing of wpQueryDefaultHelp.

wpclsQueryDefaultHelp - Usage

This method can be called at any time in order to determine the default help panel for this object class.

wpclsQueryDefaultHelp- How to Override

The default WPObject class does not process this method other than returning FALSE.

wpclsQueryDefaultHelp - Related Methods

Related Methods

- [wpDisplayHelp](#)
- [wpMenuItemHelpSelected](#)
- [wpQueryDefaultHelp](#)
- [wpSetDefaultHelp](#)
- [wpclsQueryDefaultHelp](#)

wpclsQueryDefaultHelp - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

wpclsQueryDefaultView

wpclsQueryDefaultView - Syntax

This method is called to allow the class object to specify the default open view for its instance.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */
ULONG      rv;       /* Default open view. */

rv = wpclsQueryDefaultView(somSelf);
```

wpclsQueryDefaultView Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the [ODwps](#) class object.

wpclsQueryDefaultView Return Value - rv

rv ([ULONG](#)) - returns

Default open view.

OPEN_DOCSHELL

Open document shell view.

wpclsQueryDefaultView - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

rv ([ULONG](#)) - returns
Default open view.

OPEN_DOCSHELL
Open document shell view.

wpclsQueryDefaultView - Usage

This method can be called at any time in order to query the default open view for instances of this class.

wpclsQueryDefaultView - How to Override

All classes should override this method, so that new objects in their class will always have a sensible default view (device objects typically have a default view of OPEN_SETTINGS). The default view is used for both the conditional Open cascade menu and double-clicking on the object.

wpclsQueryDefaultView - Related Methods

Related Methods

- [wpQueryDefaultView](#)
 - [wpclsQueryDefaultView](#)
 - [wpSetDefaultView](#)
-

wpclsQueryDefaultView - Topics

Class:
[ODwps](#)

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[How to Override](#)
[Usage](#)
[Related Methods](#)

wpclsQueryIconData

wpclsQueryIconData - Syntax

This class method allows the system to build the class default icon for a given class.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;    /* Pointer to the ODwps class object. */
PICONINFO  pIconInfo;   /* Pointer to the default icon for OpenDoc objects. */
ULONG      ulReturn;    /* Buffer size or number of bytes required to hold the output data. */

ulReturn = wpclsQueryIconData(somSelf, pIconInfo);
```

wpclsQueryIconData Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the ODwps class object.

wpclsQueryIconData Parameter - plconInfo

plconInfo (PICONINFO) - in/out
Pointer to the default icon for OpenDoc objects.

If this parameter is NULLHANDLE, the size should still be returned correctly.

wpclsQueryIconData Return Value - ulReturn

ulReturn (ULONG) - returns
Buffer size or number of bytes required to hold the output data.

Depending of the contents of the *plconInfo* parameter, *ulReturn* contains one of the following:

<i>plconInfo</i>	<i>ulReturn</i>
NULL	Number of bytes required to hold the output data for this class.
Not NULL	Number of bytes written into the buffer.

If *ulReturn* is 0, an error occurred.

wpclsQueryIconData - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

plconInfo ([PICONINFO](#)) - in/out
Pointer to the default icon for OpenDoc objects.

If this parameter is NULLHANDLE, the size should still be returned correctly.

ulReturn ([ULONG](#)) - returns
Buffer size or number of bytes required to hold the output data.

Depending of the contents of the *plconInfo* parameter, *ulReturn* contains one of the following:

<i>plconInfo</i>	<i>ulReturn</i>
NULL	Number of bytes required to hold the output data for this class.
Not NULL	Number of bytes written into the buffer.

If *ulReturn* is 0, an error occurred.

wpclsQueryIconData - Remarks

If NULL is passed for the *plconInfo* parameter, the caller is asking for the size of the [ICONINFO](#) buffer needed for this class (usually for memory allocation purposes). Otherwise, the *plconInfo* parameter can always be assumed to be large enough to accommodate the [ICONINFO](#) and the variable data for this class.

Note that the [ICONINFO](#) structure allows you to specify the default icon in three different ways:

- Block of binary data
- Icon file name
- Module name and resource identifier

However, only one mechanism need be supported by any given class. For example, a caller cannot request one of the three formats by prefilling the [ICONINFO](#) structure.

wpclsQueryIconData - Usage

This method may be called at any time. Typically, it would not be useful for another object class to make calls to this method.

wpclsQueryIconData - How to Override

Workplace classes that wish to have a unique class default icon must override this method and fill out the appropriate fields within the [ICONINFO](#) structure. In addition, the correct size for the [ICONINFO](#) must always be returned.

wpclsQueryIconData - Related Methods

Related Methods

- [wpQueryIcon](#)
- [wpQueryIconData](#)
- [wpSetIcon](#)
- [wpSetIconData](#)
- [wpclsQueryIcon](#)

wpclsQueryIconData - Topics

Class:

[ODwps](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

wpclsQueryInstanceFilter

wpclsQueryInstanceFilter - Syntax

This method is called to allow the class object to specify the file title filters for instances of its class.

This method is an override of the corresponding method from the WPFileSystem ancestor class.

```
#define INCL_ODWPS
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */
PSZ        psuccess; /* Success indicator. */

psuccess = wpclsQueryInstanceFilter(somSelf);
```

wpclsQueryInstanceFilter Parameter - somSelf

somSelf (ODwps *) - input

Pointer to the [ODwps](#) class object.

wpclsQueryInstanceFilter Return Value - psuccess

psuccess (PSZ) - returns
Success indicator.

NULL
Other

Error occurred.
A pointer to a string containing the "*.OD" file title filter to support the recognition of .OD files as OpenDoc files.

wpclsQueryInstanceFilter - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

psuccess (PSZ) - returns
Success indicator.

NULL
Other

Error occurred.
A pointer to a string containing the "*.OD" file title filter to support the recognition of .OD files as OpenDoc files.

wpclsQueryInstanceFilter - Remarks

It is important that the values returned by this class method are restricted to class-specific filters. For example, returning a filter of "*" could effectively make the system unstable.

wpclsQueryInstanceFilter - Usage

This method can be called at any time in order to determine which file title filters are used to determine instances of this class.

wpclsQueryInstanceFilter - How to Override

This method should be overridden in order to automatically designate file objects as instances of this class. The value returned by the override method will replace the current title filter string which is used to designate instances. If the parent method is called, it should be called first.

wpclsQueryInstanceFilter - Related Methods

Related Methods

- [wpclsQueryInstanceType](#)

wpclsQueryInstanceFilter - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

wpclsQueryInstanceType

wpclsQueryInstanceType - Syntax

This method is called to allow the class object to specify the file types for instances of its class.

This method is an override of the corresponding method from the WPFileSystem ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */
PSZ        psuccess; /* Success indicator. */

psuccess = wpclsQueryInstanceType(somSelf);
```

wpclsQueryInstanceType Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

wpclsQueryInstanceType Return Value - psuccess

psuccess ([PSZ](#)) - returns
Success indicator.

NULL
Other

Error occurred.
A pointer to a string containing "OpenDoc Document" to support the recognition of files with this .TYPE extended attribute (EA) as OpenDoc files.

wpclsQueryInstanceType - Parameters

somSelf (ODwps *) - input

Pointer to the [ODwps](#) class object.

psuccess ([PSZ](#)) - returns

Success indicator.

NULL

Other

Error occurred.

A pointer to a string containing "OpenDoc Document" to support the recognition of files with this .TYPE extended attribute (EA) as OpenDoc files.

wpclsQueryInstanceType - Remarks

It is recommended that object classes define their own special type strings.

wpclsQueryInstanceType - Usage

This method can be called at any time in order to determine which file types are used to determine instances of this class.

wpclsQueryInstanceType - How to Override

This method should be overridden in order to automatically designate file objects as instances of this class. The value returned by the override method will replace the current type string which is used to designate instances. If the parent method is called, it should be called first.

wpclsQueryInstanceType - Related Methods

Related Methods

- [wpclsQueryInstanceFilter](#)

wpclsQueryInstanceType - Topics

Class:

[ODwps](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

wpclsQueryStyle

wpclsQueryStyle - Syntax

This method is called to allow the class object to specify the default object class style for its instances.

This method is an override of the corresponding method from the WPObject ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf;    /* Pointer to the ODwps class object. */
ULONG      ulReturn;    /* Class style for this object. */

ulReturn = wpclsQueryStyle(somSelf);
```

wpclsQueryStyle Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the ODwps class object.

wpclsQueryStyle Return Value - ulReturn

ulReturn (ULONG) - returns
Class style for this object.

CLSSTYLE_DONTTEMPLATE
Do not allow a create-template operation on objects of this class.

CLSSTYLE_NEVERCOPY
Do not allow a copy operation on objects of this class.

CLSSTYLE_NEVERDELETE
Do not allow a delete operation on objects of this class.

CLSSTYLE_NEVERDRAG
Do not allow a drag operation on objects of this class.

CLSSTYLE_NEVERDROPON
Do not allow a dropon operation on objects of this class.

CLSSTYLE_NEVERLINK

Do not allow a create-shadow operation on objects of this class.

CLSSTYLE_NEVERMOVE

Do not allow a move operation on objects of this class.

CLSSTYLE_NEVERPRINT

Do not allow a print of this object.

CLSSTYLE_NEVERRENAME

Do not allow the renaming of objects of this class.

CLSSTYLE_NEVERSETTINGS

Do not allow a settings operation on objects of this class.

CLSSTYLE_NEVERVISIBLE

Make instances of this class always invisible.

CLSSTYLE_NEVERPRINT

wpclsQueryStyle - Parameters

somSelf (ODwps *) - input

Pointer to the [ODwps](#) class object.

ulReturn (ULONG) - returns

Class style for this object.

CLSSTYLE_DONTTEMPLATE

Do not allow a create-template operation on objects of this class.

CLSSTYLE_NEVERCOPY

Do not allow a copy operation on objects of this class.

CLSSTYLE_NEVERDELETE

Do not allow a delete operation on objects of this class.

CLSSTYLE_NEVERDRAG

Do not allow a drag operation on objects of this class.

CLSSTYLE_NEVERDROPON

Do not allow a dropon operation on objects of this class.

CLSSTYLE_NEVERLINK

Do not allow a create-shadow operation on objects of this class.

CLSSTYLE_NEVERMOVE

Do not allow a move operation on objects of this class.

CLSSTYLE_NEVERPRINT

Do not allow a print of this object.

CLSSTYLE_NEVERRENAME

Do not allow the renaming of objects of this class.

CLSSTYLE_NEVERSETTINGS

Do not allow a settings operation on objects of this class.

CLSSTYLE_NEVERVISIBLE

Make instances of this class always invisible.

CLSSTYLE_NEVERPRINT

wpclsQueryStyle - Remarks

When an instance is initially created, it has the same object style (OBJSTYLE_xxx) flags as its class style (CLSSTYLE_xxx).

wpclsQueryStyle - Usage

This method can be called at any time in order to determine the default style for instances of this class.

wpclsQueryStyle - How to Override

This method should be overridden in order to modify the default object style for instances of this class.

wpclsQueryStyle - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

wpclsQueryTitle

wpclsQueryTitle - Syntax

This method is called to allow the class object to specify the default title for its instances.

This method is an override of the corresponding method from the WPObj ancestor class.

```
#define INCL_WINWORKPLACE
#include <os2.h>

ODwps      *somSelf; /* Pointer to the ODwps class object. */
PSZ        rv;       /* Pointer to the national language version of the string "OpenDoc Document". */

rv = wpclsQueryTitle(somSelf);
```

wpclsQueryTitle Parameter - somSelf

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

wpclsQueryTitle Return Value - rv

rv ([PSZ](#)) - returns
Pointer to the national language version of the string "OpenDoc Document".

wpclsQueryTitle - Parameters

somSelf (ODwps *) - input
Pointer to the [ODwps](#) class object.

rv ([PSZ](#)) - returns
Pointer to the national language version of the string "OpenDoc Document".

wpclsQueryTitle - Remarks

The title is used for templates and the default for new instances of this class.

wpclsQueryTitle - Usage

This method can be called at any time in order to determine the default title for instances of this class.

wpclsQueryTitle - How to Override

All classes should override this method, so that new objects and their classes always have a sensible default title.

wpclsQueryTitle - Related Methods

Related Methods

- [wpQueryTitle](#)
- [wpclsQueryTitle](#)
- [wpSetTitle](#)

wpclsQueryTitle - Topics

Class:

ODwps

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[How to Override](#)

[Usage](#)

[Related Methods](#)

Functions and Macros

This section contains an alphabetic list of the functions and macros that can be called within OpenDoc parts, written in both C and C++ languages.

All of these functions and macros are available by including OS2.H and defining INCL_ODAPI, unless otherwise noted. The functions and macros are divided into the following subsections:

- [Byte Arrays](#)
- [International Text](#)
- [Math Routines](#)
- [OSA Descriptor Utilities](#)
- [Persistent Objects](#)
- [Registration](#)
- [Standard-Typed Values](#)
- [Templates](#)
- [Window Properties](#)

Byte Arrays

Byte arrays are used to read from and write to storage units. The functions and macros in this section are used to manipulate byte arrays, or read and write byte arrays in a storage unit or storage unit view.

The following lists the byte array functions and macros in alphabetic order.

- [AreByteArraysEqual](#)
- [CopyByteArray](#)
- [CopyByteArrayStruct](#)
- [CreateByteArray](#)
- [CreateByteArrayStruct](#)
- [CreateEmptyByteArray](#)
- [CreateEmptyByteArrayStruct](#)
- [DisposeByteArray](#)
- [DisposeByteArrayStruct](#)
- [StorageUnitGetValue](#)
- [StorageUnitSetPromiseValue](#)
- [StorageUnitSetValue](#)
- [StorageUnitViewGetValue](#)
- [StorageUnitViewSetValue](#)
- [UseByteArray](#)

AreByteArraysEqual

AreByteArraysEqual - Syntax

This function tests if two byte arrays are equal.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray    *ba1;
ODByteArray    *ba2;
ODBoolean      rc;

rc = AreByteArraysEqual(ba1, ba2);
```

AreByteArraysEqual Parameter - ba1

ba1 (ODByteArray *) - input
The first byte array to be compared.

AreByteArraysEqual Parameter - ba2

ba2 (ODByteArray *) - input
The second byte array to be compared.

AreByteArraysEqual Return Value - rc

rc (ODBoolean) - returns
A flag indicating whether the byte arrays are equal in content and size.

kODTrue	The byte arrays are equal.
kODFalse	The byte arrays are not equal.

AreByteArraysEqual - Parameters

ba1 (ODByteArray *) - input
The first byte array to be compared.

ba2 ([ODByteArray *](#)) - input
The second byte array to be compared.

rc ([ODBoolean](#)) - returns
A flag indicating whether the byte arrays are equal in content and size.

kODTrue	The byte arrays are equal.
kODFalse	The byte arrays are not equal.

AreByteArraysEqual - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

CopyByteArray

CopyByteArray - Syntax

This function copies a byte array and returns a pointer to that copy.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *fromBA;
ODByteArray      *rc;

rc = CopyByteArray(fromBA);
```

CopyByteArray Parameter - fromBA

fromBA ([ODByteArray *](#)) - input
The byte array to be copied.

CopyByteArray Return Value - rc

rc ([ODByteArray *](#)) - returns

A pointer to a copy of the specified byte array.

CopyByteArray - Parameters

fromBA ([ODByteArray *](#)) - input
The byte array to be copied.

rc ([ODByteArray *](#)) - returns
A pointer to a copy of the specified byte array.

CopyByteArray - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

CopyByteArrayStruct

CopyByteArrayStruct - Syntax

This function copies a buffer into a byte array.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *fromBA;
ODByteArray      rc;

rc = CopyByteArrayStruct(fromBA);
```

CopyByteArrayStruct Parameter - fromBA

fromBA ([ODByteArray *](#)) - input
The byte array to be copied.

CopyByteArrayStruct Return Value - rc

rc ([ODByteArray](#)) - returns
A copy of the specified byte array.

CopyByteArrayStruct - Parameters

fromBA ([ODByteArray *](#)) - input
The byte array to be copied.

rc ([ODByteArray](#)) - returns
A copy of the specified byte array.

CopyByteArrayStruct - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

CreateByteArray

CreateByteArray - Syntax

This function creates a byte array from a buffer of a specified length.

```
#define INCL_ODAPI
#include <os2.h>

void          *buffer;
ODULong      size;
ODByteArray  *rc;

rc = CreateByteArray(buffer, size);
```

CreateByteArray Parameter - buffer

buffer - input
A buffer containing data to be copied into the byte array.

CreateByteArray Parameter - size

size ([ODULong](#)) - input
The size of the specified buffer.

CreateByteArray Return Value - rc

rc ([ODByteArray *](#)) - returns
A pointer to the new byte array.

CreateByteArray - Parameters

buffer - input
A buffer containing data to be copied into the byte array.

size ([ODULong](#)) - input
The size of the specified buffer.

rc ([ODByteArray *](#)) - returns
A pointer to the new byte array.

CreateByteArray - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

CreateByteArrayStruct

CreateByteArrayStruct - Syntax

This function creates a byte array from a buffer of a specified length.

```
#define INCL_ODAPI
```



```
#include <os2.h>

void          *buffer;
ODULong       size;
ODByteArray   rc;

rc = CreateByteArrayStruct (buffer, size);
```

CreateByteArrayStruct Parameter - buffer

buffer - input
A buffer containing data to be copied into the byte array.

CreateByteArrayStruct Parameter - size

size ([ODULong](#)) - input
The size of the specified buffer.

CreateByteArrayStruct Return Value - rc

rc ([ODByteArray](#)) - returns
A new byte array.

CreateByteArrayStruct - Parameters

buffer - input
A buffer containing data to be copied into the byte array.

size ([ODULong](#)) - input
The size of the specified buffer.

rc ([ODByteArray](#)) - returns
A new byte array.

CreateByteArrayStruct - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

CreateEmptyByteArray

CreateEmptyByteArray - Syntax

This function creates an empty byte array.

```
#define INCL_ODAPI
#include <os2.h>

ODULong      maximum;
ODByteArray  *rc;

rc = CreateEmptyByteArray(maximum);
```

CreateEmptyByteArray Parameter - maximum

maximum (ODULong) - input
The maximum size of the byte array to be created.

CreateEmptyByteArray Return Value - rc

rc (ODByteArray *) - returns
A pointer to the new byte array.

CreateEmptyByteArray - Parameters

maximum (ODULong) - input
The maximum size of the byte array to be created.

rc (ODByteArray *) - returns
A pointer to the new byte array.

CreateEmptyByteArray - Topics

Select an item:

CreateEmptyByteArrayStruct

CreateEmptyByteArrayStruct - Syntax

This function creates an empty byte array.

```
#define INCL_ODAPI
#include <os2.h>

ODULong      maximum;
ODByteArray  rc;

rc = CreateEmptyByteArrayStruct (maximum);
```

CreateEmptyByteArrayStruct Parameter - maximum

maximum (ODULong) - input
The maximum size of the byte array to be created.

CreateEmptyByteArrayStruct Return Value - rc

rc (ODByteArray) - returns
A new byte array.

CreateEmptyByteArrayStruct - Parameters

maximum (ODULong) - input
The maximum size of the byte array to be created.

rc (ODByteArray) - returns
A new byte array.

CreateEmptyByteArrayStruct - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

DisposeByteArray

DisposeByteArray - Syntax

This function deletes the specified byte array and its content.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *byteArray;

DisposeByteArray (byteArray);
```

DisposeByteArray Parameter - byteArray

byteArray ([ODByteArray](#) *) - input
A pointer to the byte array to be deleted.

DisposeByteArray - Return Value

None.

DisposeByteArray - Parameters

byteArray ([ODByteArray](#) *) - input
A pointer to the byte array to be deleted.

None.

DisposeByteArray - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

DisposeByteArrayStruct

DisposeByteArrayStruct - Syntax

This function deletes the specified byte array and its content.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *byteArray;

DisposeByteArrayStruct (byteArray);
```

DisposeByteArrayStruct Parameter - byteArray

byteArray ([ODByteArray](#) *) - input
A pointer to the byte array to be deleted.

DisposeByteArrayStruct - Return Value

None.

DisposeByteArrayStruct - Parameters

byteArray ([ODByteArray](#) *) - input
A pointer to the byte array to be deleted.

None.

DisposeByteArrayStruct - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

StorageUnitGetValue

StorageUnitGetValue - Syntax

This function reads a value from a storage unit, starting at the offset.

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit    *su;
Environment      *ev;
ODULong          size;
ODPtr            buffer;
ODULong          rc;

rc = StorageUnitGetValue(su, ev, size, buffer);
```

StorageUnitGetValue Parameter - su

su (ODStorageUnit *) - input

The storage unit whose value is to be read. This storage unit is prefocussed to the appropriate property and value.

StorageUnitGetValue Parameter - ev

ev (Environment *) - input

The SOM environment.

StorageUnitGetValue Parameter - size

size ([ODULong](#)) - input
The size of the buffer.

StorageUnitGetValue Parameter - buffer

buffer ([ODPtr](#)) - input
The buffer to read data into.

StorageUnitGetValue Return Value - rc

rc ([ODULong](#)) - returns
The number of bytes read.

StorageUnitGetValue - Parameters

su ([ODStorageUnit *](#)) - input
The storage unit whose value is to be read. This storage unit is prefocussed to the appropriate property and value.

ev ([Environment *](#)) - input
The SOM environment.

size ([ODULong](#)) - input
The size of the buffer.

buffer ([ODPtr](#)) - input
The buffer to read data into.

rc ([ODULong](#)) - returns
The number of bytes read.

StorageUnitGetValue - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

StorageUnitSetPromiseValue

StorageUnitSetPromiseValue - Syntax

This macro writes a promise into a storage unit, starting at the offset.

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit    *su;
Environment      *ev;
ODULong          type;
ODULong          offset;
ODULong          size;
void             *buffer;
ODPart           sourcePart;

StorageUnitSetPromiseValue(su, ev, type, offset,
                           size, buffer, sourcePart);
```

StorageUnitSetPromiseValue Parameter - su

su (ODStorageUnit *) - input

The storage unit to write the promise into. This storage unit is prefocussed to the appropriate property and value.

StorageUnitSetPromiseValue Parameter - ev

ev (Environment *) - input

The SOM environment.

StorageUnitSetPromiseValue Parameter - type

type (ODULong) - input

The type of value to contain the promise.

StorageUnitSetPromiseValue Parameter - offset

offset (ODULong) - input

The offset at which the promise data is to be written in the storage unit.

StorageUnitSetPromiseValue Parameter - size

size ([ODULong](#)) - input
The size of the promise to be written.

StorageUnitSetPromiseValue Parameter - buffer

buffer - input
A pointer to a buffer containing the promise data.

StorageUnitSetPromiseValue Parameter - sourcePart

sourcePart ([ODPart](#)) - input
A reference to the part that made the promise.

StorageUnitSetPromiseValue - Return Value

None.

StorageUnitSetPromiseValue - Parameters

su ([ODStorageUnit *](#)) - input
The storage unit to write the promise into. This storage unit is prefocussed to the appropriate property and value.

ev ([Environment *](#)) - input
The SOM environment.

type ([ODULong](#)) - input
The type of value to contain the promise.

offset ([ODULong](#)) - input
The offset at which the promise data is to be written in the storage unit.

size ([ODULong](#)) - input
The size of the promise to be written.

buffer - input
A pointer to a buffer containing the promise data.

sourcePart ([ODPart](#)) - input
A reference to the part that made the promise.

None.

StorageUnitSetPromiseValue - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

StorageUnitSetValue

StorageUnitSetValue - Syntax

This macro writes a value into a storage unit, starting at the offset.

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit    *su;
Environment      *ev;
ODULong          size;
void             *buffer;

StorageUnitSetValue(su, ev, size, buffer);
```

StorageUnitSetValue Parameter - su

su (ODStorageUnit *) - input

The storage unit to write the value into. This storage unit is prefocussed to the appropriate property and value.

StorageUnitSetValue Parameter - ev

ev (Environment *) - input

The SOM environment.

StorageUnitSetValue Parameter - size

size ([ODULong](#)) - input
The size of the buffer.

StorageUnitSetValue Parameter - buffer

buffer - input
A pointer to the buffer containing the value to be written.

StorageUnitSetValue - Return Value

None.

StorageUnitSetValue - Parameters

su ([ODStorageUnit *](#)) - input
The storage unit to write the value into. This storage unit is prefocussed to the appropriate property and value.

ev ([Environment *](#)) - input
The SOM environment.

size ([ODULong](#)) - input
The size of the buffer.

buffer - input
A pointer to the buffer containing the value to be written.

None.

StorageUnitSetValue - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

StorageUnitViewGetValue

StorageUnitViewGetValue - Syntax

This function reads data from the currently focused value in the specified storage unit, starting at the offset.

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *suv;
Environment          *ev;
ODULong              size;
ODPtr                buffer;
ODULong              rc;

rc = StorageUnitViewGetValue(suv, ev, size,
                             buffer);
```

StorageUnitViewGetValue Parameter - suv

suv (ODStorageUnitView *) - input
The storage unit to be queried.

StorageUnitViewGetValue Parameter - ev

ev (Environment *) - input
The SOM environment.

StorageUnitViewGetValue Parameter - size

size (ODULong) - input
The size of the buffer.

StorageUnitViewGetValue Parameter - buffer

buffer (ODPtr) - input
A pointer to a buffer to read data into.

StorageUnitViewGetValue Return Value - rc

rc ([ODULong](#)) - returns
The number of bytes read.

StorageUnitViewGetValue - Parameters

suv ([ODStorageUnitView *](#)) - input
The storage unit to be queried.

ev ([Environment *](#)) - input
The SOM environment.

size ([ODULong](#)) - input
The size of the buffer.

buffer ([ODPtr](#)) - input
A pointer to a buffer to read data into.

rc ([ODULong](#)) - returns
The number of bytes read.

StorageUnitViewGetValue - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

StorageUnitViewSetValue

StorageUnitViewSetValue - Syntax

This macro writes data to the currently focused value in the specified storage unit, starting at the offset.

```
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *suv;
Environment           *ev;
ODULong              size;
void                 *buffer;

StorageUnitViewSetValue(suv, ev, size, buffer);
```

StorageUnitViewSetValue Parameter - suv

suv (ODStorageUnitView *) - output
The storage unit to be queried.

StorageUnitViewSetValue Parameter - ev

ev (Environment *) - input
The SOM environment.

StorageUnitViewSetValue Parameter - size

size (ODULong) - input
The size of the buffer.

StorageUnitViewSetValue Parameter - buffer

buffer - input
A pointer to the buffer containing the data to be written.

StorageUnitViewSetValue - Return Value

None.

StorageUnitViewSetValue - Parameters

suv (ODStorageUnitView *) - output
The storage unit to be queried.

ev (Environment *) - input
The SOM environment.

size (ODULong) - input
The size of the buffer.

buffer - input

A pointer to the buffer containing the data to be written.

None.

StorageUnitViewSetValue - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

UseByteArray

UseByteArray - Syntax

This function instantiates a byte array to the specified buffer of a specified size.

```
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *ba;
void             *buffer;
ODULong          size;

UseByteArray(ba, buffer, size);
```

UseByteArray Parameter - ba

ba ([ODByteArray *](#)) - input

The byte array to be instantiated.

UseByteArray Parameter - buffer

buffer - input

A buffer containing the data.

UseByteArray Parameter - size

size ([ODULong](#)) - input
The size of the specified buffer.

UseByteArray - Return Value

None.

UseByteArray - Parameters

ba ([ODByteArray *](#)) - input
The byte array to be instantiated.

buffer - input
A buffer containing the data.

size ([ODULong](#)) - input
The size of the specified buffer.

None.

UseByteArray - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

International Text

This sections contains functions and macros used by part handlers for manipulating international text (IText) structures.

The following lists the IText functions and macros in alphabetic order.

- [CopyIText](#)
- [CopyITextStruct](#)
- [CreateIText](#)
- [CreateITextClear](#)
- [CreateITextCString](#)
- [CreateITextPString](#)
- [DisposeIText](#)

- [DisposeITextStruct](#)
- [GetCStringFromIText](#)
- [GetPStringFromIText](#)
- [GetITextCodePage](#)
- [GetITextCString](#)
- [GetITextLangCode](#)
- [GetITextPString](#)
- [GetITextPStringFromIText](#)
- [GetITextPtr](#)
- [GetITextStringLength](#)
- [SetITextBufferSize](#)
- [SetITextCodePage](#)
- [SetITextCString](#)
- [SetITextLangCode](#)
- [SetITextPString](#)
- [SetITextStringLength](#)
- [SetITextText](#)

CopyIText

CopyIText - Syntax

This function creates a copy of the specified IText structure.

```
#include <os2.h>

ODIText      *original;
ODIText      *rc;

rc = CopyIText(original);
```

CopyIText Parameter - original

original ([ODIText *](#)) - input
The IText structure to be copied.

CopyIText Return Value - rc

rc ([ODIText *](#)) - returns
A pointer to the copy of the specified IText structure.

CopyIText - Parameters

original ([ODIText *](#)) - input
The IText structure to be copied.

rc ([ODIText *](#)) - returns
A pointer to the copy of the specified IText structure.

CopyIText - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

CopyITextStruct

CopyITextStruct - Syntax

This function creates a copy of the specified IText structure.

```
#include <os2.h>

ODIText      *original;
ODIText      *rc;

rc = CopyITextStruct(original);
```

CopyITextStruct Parameter - original

original ([ODIText *](#)) - input
A pointer to the IText structure to be copied.

CopyITextStruct Return Value - rc

rc ([ODIText *](#)) - returns
A copy of the specified IText structure.

CopyITTextStruct - Parameters

original ([ODIText *](#)) - input
A pointer to the IText structure to be copied.

rc ([ODIText *](#)) - returns
A copy of the specified IText structure.

CopyITTextStruct - Topics

Select an item:

- [Syntax](#)
- [Parameters](#)
- [Returns](#)

CreateIText

CreateIText - Syntax

This function creates an IText structure from the specified buffer of characters.

```
#include <os2.h>

ODCodePage    theCodePage;
ODLangCode    theLangCode;
ODUByte       *text;
ODSize        *textLength;
ODIText       *rc;

rc = CreateIText(theCodePage, theLangCode,
                text, textLength);
```

CreateIText Parameter - theCodePage

theCodePage ([ODCodePage](#)) - input
The code page; the default is 0.

CreateIText Parameter - theLangCode

theLangCode ([ODLangCode](#)) - input
The language code; the default is 0.

CreateIText Parameter - text

text ([ODUByte *](#)) - input
A buffer of characters from which the IText is to be created.

CreateIText Parameter - textLength

textLength ([ODSize *](#)) - input
The size, in bytes, of the buffer.

CreateIText Return Value - rc

rc ([ODIText *](#)) - returns
A pointer to the new IText structure.

CreateIText - Parameters

theCodePage ([ODCodePage](#)) - input
The code page; the default is 0.

theLangCode ([ODLangCode](#)) - input
The language code; the default is 0.

text ([ODUByte *](#)) - input
A buffer of characters from which the IText is to be created.

textLength ([ODSize *](#)) - input
The size, in bytes, of the buffer.

rc ([ODIText *](#)) - returns
A pointer to the new IText structure.

CreateIText - Remarks

This function is used for text that is neither a C string or a Pascal string.

CreateIText - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

CreateITextClear

CreateITextClear - Syntax

This function creates an IText structure of a specified length.

```
#include <os2.h>

ODCodePage    theCodePage;
ODLangCode    theLangCode;
ODSize        stringLength;
ODIText       *rc;

rc = CreateITextClear(theCodePage, theLangCode,
                     stringLength);
```

CreateITextClear Parameter - theCodePage

theCodePage ([ODCodePage](#)) - input
The code page; the default is 0.

CreateITextClear Parameter - theLangCode

theLangCode ([ODLangCode](#)) - input
The language code; the default is 0.

CreateITextClear Parameter - stringLength

stringLength ([ODSize](#)) - input
The size, in bytes, of the IText structure.

CreateITextClear Return Value - rc

rc ([ODIText *](#)) - returns
A pointer to the new IText structure.

CreateITextClear - Parameters

theCodePage ([ODCodePage](#)) - input
The code page; the default is 0.

theLangCode ([ODLangCode](#)) - input
The language code; the default is 0.

stringLength ([ODSize](#)) - input
The size, in bytes, of the IText structure.

rc ([ODIText *](#)) - returns
A pointer to the new IText structure.

CreateITextClear - Remarks

There is an overloaded C++ function call, [CreateIText](#) that performs the same function.

CreateITextClear - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

CreateITextCString

CreateITextCString - Syntax

This function creates an IText structure from the specified null-terminated string.

```
#include <os2.h>

ODCodePage    theCodePage;
ODLangCode    theLangCode;
char          *text;
ODIText       *rc;

rc = CreateITextCString(theCodePage, theLangCode,
                       text);
```

CreateITextCString Parameter - theCodePage

theCodePage (ODCodePage) - input
The code page; the default is 0.

CreateITextCString Parameter - theLangCode

theLangCode (ODLangCode) - input
The language code. The language code is the code page of the characters encoded in the character string.

CreateITextCString Parameter - text

text (char *) - input
A null-terminated character string.

CreateITextCString Return Value - rc

rc (ODIText *) - returns
A pointer to the new IText structure.

CreateITextCString - Parameters

theCodePage (ODCodePage) - input
The code page; the default is 0.

theLangCode (ODLangCode) - input
The language code. The language code is the code page of the characters encoded in the character string.

text (char *) - input
A null-terminated character string.

rc ([ODIText *](#)) - returns
A pointer to the new IText structure.

CreateITextCString - Remarks

There is an overloaded C++ function call, [CreateIText](#) that performs the same function.

CreateITextCString - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

CreateITextPString

CreateITextPString - Syntax

This function creates an IText structure from a specified array of 255 characters.

```
#include <os2.h>

ODCodePage    theCodePage;
ODLangCode    theLangCode;
StringPtr     text;
ODIText       *rc;

rc = CreateITextPString(theCodePage, theLangCode,
                       text);
```

CreateITextPString Parameter - theCodePage

theCodePage ([ODCodePage](#)) - input
The code page; the default is 0.

CreateITextPString Parameter - theLangCode

theLangCode ([ODLangCode](#)) - input

The language code. The language code is the code page of the characters encoded in the character string.

CreateITextPString Parameter - text

text (StringPtr) - input

A pointer to an array of 255 characters. The first byte of this array contains its length.

CreateITextPString Return Value - rc

rc ([ODIText *](#)) - returns

A pointer to the new IText structure.

CreateITextPString - Parameters

theCodePage ([ODCodePage](#)) - input

The code page; the default is 0.

theLangCode ([ODLangCode](#)) - input

The language code. The language code is the code page of the characters encoded in the character string.

text (StringPtr) - input

A pointer to an array of 255 characters. The first byte of this array contains its length.

rc ([ODIText *](#)) - returns

A pointer to the new IText structure.

CreateITextPString - Remarks

These is an overloaded C++ function call, [CreateIText](#) that performs the same function.

CreateITextPString - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

DisposeIText

DisposeIText - Syntax

This function frees memory of an IText structure.

```
#include <os2.h>

ODIText      *iText;

DisposeIText (iText);
```

DisposeIText Parameter - iText

iText ([ODIText *](#)) - input
The IText structure to be freed.

DisposeIText - Return Value

None.

DisposeIText - Parameters

iText ([ODIText *](#)) - input
The IText structure to be freed.

None.

DisposeIText - Topics

Select an item:

[Syntax](#)
[Parameters](#)

DisposeITextStruct

DisposeITextStruct - Syntax

This macro frees the memory of an IText structure.

```
#include <os2.h>

ODIText      *iText;

DisposeITextStruct(iText);
```

DisposeITextStruct Parameter - iText

iText (ODIText *) - input
The IText structure to be freed.

DisposeITextStruct - Return Value

None.

DisposeITextStruct - Parameters

iText (ODIText *) - input
The IText structure to be freed.

None.

DisposeITextStruct - Topics

Select an item:

GetCStringFromIText

GetCStringFromIText - Syntax

This function returns a pointer to a null-terminated character string in an IText structure.

```
#include <os2.h>

ODIText      *iText;
char         *rc;

rc = GetCStringFromIText(iText);
```

GetCStringFromIText Parameter - iText

iText ([ODIText *](#)) - input
The IText structure to be queried.

GetCStringFromIText Return Value - rc

rc (char *) - returns
The null-terminated string of the specified IText structure.

GetCStringFromIText - Parameters

iText ([ODIText *](#)) - input
The IText structure to be queried.

rc (char *) - returns
The null-terminated string of the specified IText structure.

GetCStringFromIText - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetPStringFromIText

GetPStringFromIText - Syntax

This function returns an array of characters from the specified IText structure.

```
#include <os2.h>

ODIText      *iText;
StringPtr     rc;

rc = GetPStringFromIText(iText);
```

GetPStringFromIText Parameter - iText

iText ([ODIText *](#)) - input
The IText structure to be queried.

GetPStringFromIText Return Value - rc

rc (StringPtr) - returns
An array of characters from the specified IText structure. The first byte in this array contains the length of the string.

GetPStringFromIText - Parameters

iText ([ODIText *](#)) - input
The IText structure to be queried.

rc (StringPtr) - returns
An array of characters from the specified IText structure. The first byte in this array contains the length of the string.

GetPStringFromIText - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetITextCodePage

GetITextCodePage - Syntax

This function returns the code page for the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODCodePage    theCodePage;

theCodePage = GetITextCodePage(text);
```

GetITextCodePage Parameter - text

text ([ODIText *](#)) - input
The IText structure to be queried.

GetITextCodePage Return Value - theCodePage

theCodePage ([ODCodePage](#)) - returns
The code page of the specified IText structure.

GetITextCodePage - Parameters

text ([ODIText *](#)) - input
The IText structure to be queried.

theCodePage ([ODCodePage](#)) - returns

The code page of the specified IText structure.

GetITextCodePage - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetITextCString

GetITextCString - Syntax

This function returns a pointer to the null-terminated character string of the specified IText structure.

```
#include <os2.h>

ODIText      *text;
char         *cstring;
char         *rc;

rc = GetITextCString(text, cstring);
```

GetITextCString Parameter - text

text ([ODIText](#) *) - input
The IText structure to be queried.

GetITextCString Parameter - cstring

cstring (char *) - input
The null-terminated character string of the specified IText structure.

GetITextCString Return Value - rc

rc (char *) - returns
The null-terminated character string of the specified IText structure.

GetITextCString - Parameters

text ([ODIText](#) *) - input
The IText structure to be queried.

cstring (char *) - input
The null-terminated character string of the specified IText structure.

rc (char *) - returns
The null-terminated character string of the specified IText structure.

GetITextCString - Remarks

There is are overloaded C++ function calls, [GetITextString](#) and [GetCStringFromIText](#), [GetCStringFromIText](#), that perform the same function.

GetITextCString - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

GetITextLangCode

GetITextLangCode - Syntax

This function returns the language code for the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODILangCode   theLangCode;

theLangCode = GetITextLangCode(text);
```

GetITextLangCode Parameter - text

text ([ODIText *](#)) - input
The IText structure to be queried.

GetITextLangCode Return Value - theLangCode

theLangCode ([ODLangCode](#)) - returns
The language code of the specified IText structure.

GetITextLangCode - Parameters

text ([ODIText *](#)) - input
The IText structure to be queried.

theLangCode ([ODLangCode](#)) - returns
The language code of the specified IText structure.

GetITextLangCode - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

GetITextPString

GetITextPString - Syntax

This function returns the array of characters from the specified IText structure.

```
#include <os2.h>

ODIText      *iText;
Str255       pstring;
char         *rc;

rc = GetITextPString(iText, pstring);
```

GetITextPString Parameter - iText

iText ([ODIText *](#)) - input
The IText structure to be queried.

GetITextPString Parameter - pstring

pstring ([Str255](#)) - input
If this parameter contains a valid string, the text from the *iText* parameter is copied into this character string, and a pointer to this string is returned. If this parameter is kODNULL, a new string is allocated, the text from the *iText* parameter is copied into this character string, and a pointer to this string is returned.

GetITextPString Return Value - rc

rc (char *) - returns
An array of characters from the specified IText structure. The first byte in this array contains the length of the string.

GetITextPString - Parameters

iText ([ODIText *](#)) - input
The IText structure to be queried.

pstring ([Str255](#)) - input
If this parameter contains a valid string, the text from the *iText* parameter is copied into this character string, and a pointer to this string is returned. If this parameter is kODNULL, a new string is allocated, the text from the *iText* parameter is copied into this character string, and a pointer to this string is returned.

rc (char *) - returns
An array of characters from the specified IText structure. The first byte in this array contains the length of the string.

GetITextPString - Remarks

There is are overloaded C++ function calls, `GetITextString` and [GetCStringFromIText](#), that perform the same function.

GetITextPString - Topics

Select an item:
[Syntax](#)

GetITextPStringFromIText

GetITextPStringFromIText - Syntax

This function returns an array of characters from the specified IText structure.

```
#include <os2.h>

ODIText      *iText;
StringPtr     pstring;

pstring = GetITextPStringFromIText(iText);
```

GetITextPStringFromIText Parameter - iText

iText (ODIText *) - input
The IText structure to be queried.

GetITextPStringFromIText Return Value - pstring

pstring (StringPtr) - returns
The character array of the specified IText structure. The first byte in the array contains the length of the string.

GetITextPStringFromIText - Parameters

iText (ODIText *) - input
The IText structure to be queried.

pstring (StringPtr) - returns
The character array of the specified IText structure. The first byte in the array contains the length of the string.

GetITextPStringFromIText - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetITextPtr

GetITextPtr - Syntax

This function returns a pointer to a byte array in the IText structure.

```
#include <os2.h>

ODIText      *text;
char         *ptr;

ptr = GetITextPtr(text);
```

GetITextPtr Parameter - text

text ([ODIText *](#)) - input
The IText structure to be queried.

GetITextPtr Return Value - ptr

ptr (char *) - returns
A pointer to the byte array of the specified IText structure.

GetITextPtr - Parameters

text ([ODIText *](#)) - input
The IText structure to be queried.

ptr (char *) - returns
A pointer to the byte array of the specified IText structure.

GetITextPtr - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

GetITextStringLength

GetITextStringLength - Syntax

This function returns the length of the character string in the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODULong      length;

length = GetITextStringLength(text);
```

GetITextStringLength Parameter - text

text ([ODIText *](#)) - input
The IText structure to be queried.

GetITextStringLength Return Value - length

length ([ODULong](#)) - returns
The length of the character string in the specified IText structure.

GetITextStringLength - Parameters

text ([ODIText *](#)) - input
The IText structure to be queried.

length ([ODULong](#)) - returns

The length of the character string in the specified IText structure.

GetITextStringLength - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

SetITextBufferSize

SetITextBufferSize - Syntax

This function sets the size of the byte-array buffer in the specified IText structure. If the input values are null, this function creates and returns a new [ODIText](#) structure.

```
#include <os2.h>

ODIText      *text;
ODSize       bufferSize;
ODBoolean     preserveContents;
ODIText      *rc;

rc = SetITextBufferSize(text, bufferSize,
                        preserveContents);
```

SetITextBufferSize Parameter - text

text ([ODIText *](#)) - input

A pointer to the IText structure to be modified.

SetITextBufferSize Parameter - bufferSize

bufferSize ([ODSize](#)) - input

The new size of the byte array.

SetITextBufferSize Parameter - preserveContents

preserveContents ([ODBoolean](#)) - input
A flag indicating whether to preserve the content of the byte array, up to the length specified in the *bufferSize* parameter.

kODTrue	The content of the byte array should be preserved.
kODFalse	The content of the byte array should not be preserved.

SetITextBufferSize Return Value - rc

rc ([ODIText *](#)) - returns
A pointer to the IText structure with the new buffer size.

SetITextBufferSize - Parameters

text ([ODIText *](#)) - input
A pointer to the IText structure to be modified.

bufferSize ([ODSize](#)) - input
The new size of the byte array.

preserveContents ([ODBoolean](#)) - input
A flag indicating whether to preserve the content of the byte array, up to the length specified in the *bufferSize* parameter.

kODTrue	The content of the byte array should be preserved.
kODFalse	The content of the byte array should not be preserved.

rc ([ODIText *](#)) - returns
A pointer to the IText structure with the new buffer size.

SetITextBufferSize - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

SetITextCodePage

SetITextCodePage - Syntax

This function sets the code page for the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODCodePage   theCodePage;

SetITextCodePage(text, theCodePage);
```

SetITextCodePage Parameter - text

text ([ODIText *](#)) - input
The IText structure to be modified.

SetITextCodePage Parameter - theCodePage

theCodePage ([ODCodePage](#)) - input
The code page to be set for the specified IText structure..

SetITextCodePage - Return Value

None.

SetITextCodePage - Parameters

text ([ODIText *](#)) - input
The IText structure to be modified.

theCodePage ([ODCodePage](#)) - input
The code page to be set for the specified IText structure..

None.

SetITextCodePage - Topics

Select an item:

SetITextCString

SetITextCString - Syntax

This function sets the value of the IText structure from the specified null-terminated character string. The byte array in the IText structure is automatically extended to fit the string.

```
#include <os2.h>

ODIText      *itext;
char         *text;

SetITextCString(itext, text);
```

SetITextCString Parameter - itext

itext (ODIText *) - input
The IText structure to be modified.

SetITextCString Parameter - text

text (char *) - input
The null-terminated character string to be copied into the specified IText structure.

SetITextCString - Return Value

None

SetITextCString - Parameters

itext ([ODIText](#) *) - input
The IText structure to be modified.

text (char *) - input
The null-terminated character string to be copied into the specified IText structure.

None

SetITextCString - Remarks

There is an overloaded C++ function call, SetITextString, that performs the same function.

SetITextCString - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetITextLangCode

SetITextLangCode - Syntax

This function sets the language code for the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODLangCode   theLangCode;

SetITextLangCode(text, theLangCode);
```

SetITextLangCode Parameter - text

text ([ODIText](#) *) - input
The IText structure to be modified.

SetITextLangCode Parameter - theLangCode

theLangCode ([ODLangCode](#)) - input

The language code to be set for the specified IText structure.

SetITextLangCode - Return Value

None.

SetITextLangCode - Parameters

text ([ODIText *](#)) - input

The IText structure to be modified.

theLangCode ([ODLangCode](#)) - input

The language code to be set for the specified IText structure.

None.

SetITextLangCode - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

SetITextPString

SetITextPString - Syntax

This function sets the value of the IText structure to the specified array of 255 characters.

```
#include <os2.h>
```

```
ODIText      *itext;
```

```
StringPtr    text;  
SetITextPString(itext, text);
```

SetITextPString Parameter - itext

itext ([ODIText *](#)) - input
The IText structure to be modified.

SetITextPString Parameter - text

text (StringPtr) - input
An array of 255 characters to be set in the specified IText structure.

SetITextPString - Return Value

None.

SetITextPString - Parameters

itext ([ODIText *](#)) - input
The IText structure to be modified.

text (StringPtr) - input
An array of 255 characters to be set in the specified IText structure.

None.

SetITextPString - Remarks

There is an overloaded C++ function call, SetITextString, that performs the same function.

SetITextPString - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetITextStringLength

SetITextStringLength - Syntax

This function sets the length of the specified IText structure.

```
#include <os2.h>

ODIText      *text;
ODSize       length;
ODBoolean     preserveText;
ODIText      *rc;

rc = SetITextStringLength(text, length, preserveText);
```

SetITextStringLength Parameter - text

text ([ODIText](#) *) - input

The IText structure text to be modified. If this parameter is null, this function allocates and returns a new [ODIText](#) structure.

SetITextStringLength Parameter - length

length ([ODSize](#)) - input

The new length of the IText structure.

SetITextStringLength Parameter - preserveText

preserveText ([ODBoolean](#)) - input

A flag indicating whether to preserve the content of the byte array up to the length specified in the *length* parameter.

kODTrue	The content of the byte array should be preserved.
kODFalse	The content of the byte array should not be preserved.

SetITextStringLength Return Value - rc

rc ([ODIText *](#)) - returns
A pointer to the IText structure with the new buffer size.

SetITextStringLength - Parameters

text ([ODIText *](#)) - input
The IText structure text to be modified. If this parameter is null, this function allocates and returns a new [ODIText](#) structure.

length ([ODSize](#)) - input
The new length of the IText structure.

preserveText ([ODBoolean](#)) - input
A flag indicating whether to preserve the content of the byte array up to the length specified in the *length* parameter.

kODTrue	The content of the byte array should be preserved.
kODFalse	The content of the byte array should not be preserved.

rc ([ODIText *](#)) - returns
A pointer to the IText structure with the new buffer size.

SetITextStringLength - Remarks

There is an overloaded C++ function call, `CreateIttext`, that performs the same function.

SetITextStringLength - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

SetITextText

SetITextText - Syntax

This function sets the value of the IText structure to the specified buffer. The byte array of the IText structure is automatically extended to fit the buffer.

```
#include <os2.h>

ODIText      *itext;
ODUByte      *text;
ODSize       textLength;

SetITextText(itext, text, textLength);
```

SetITextText Parameter - itext

itext (ODIText *) - input
The IText structure to be modified.

SetITextText Parameter - text

text (ODUByte *) - input
The buffer of characters to be set in the specified IText structure..

SetITextText Parameter - textLength

textLength (ODSize) - input
The size of the buffer specified in the *text* parameter.

SetITextText - Return Value

None.

SetITextText - Parameters

itext (ODIText *) - input
The IText structure to be modified.

text (ODUByte *) - input

The buffer of characters to be set in the specified IText structure..

textLength ([ODSize](#)) - input

The size of the buffer specified in the *text* parameter.

None.

SetITextText - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

Math Routines

This section contains functions and macros to perform math operations and conversions for the following types:

ODFixed	A 32-bit fixed-point value used to represent non-integer numbers in the range [-32768, 32768). The most-significant 16 bits are integral and the least-significant 16 bits are fractional.
ODFract	A 32-bit fixed-point value used to represent non-integer numbers in the range [-2, 2). The most-significant 2 bits are integral and the least-significant 30 bits are fractional.
ODFloat	A floating-point value.
ODWide	A 64-bit signed integer.

The following lists the byte array functions and macros in alphabetic order.

- [ODFirstBit](#)
- [ODFixedDivide](#)
- [ODFixedRound](#)
- [ODFixedMultiply](#)
- [ODFixedToFloat](#)
- [ODFixedToFract](#)
- [ODFloatToFixed](#)
- [ODFloatToFract](#)
- [ODFractDivide](#)
- [ODFractMultiply](#)
- [ODFractSinCos](#)
- [ODFractToFixed](#)
- [ODFractToFloat](#)
- [IntToODFixed](#)
- [ODWideAdd](#)
- [ODWideCompare](#)
- [ODWideDivide](#)
- [ODWideIsLong](#)
- [ODWideMultiply](#)
- [ODWideNegate](#)
- [ODWideShift](#)
- [ODWideSquareRoot](#)
- [ODWideSubtract](#)

ODFirstBit

ODFirstBit - Syntax

This function returns the position of the first bit set in a signed long.

```
#include <odmath.h>

ODSLong    a;
ODSShort    rc;

rc = ODFirstBit(a);
```

ODFirstBit Parameter - a

a ([ODSLong](#)) - input
The signed long to be queried.

ODFirstBit Return Value - rc

rc ([ODSShort](#)) - returns
The position of the first bit set in the specified signed long.

ODFirstBit - Parameters

a ([ODSLong](#)) - input
The signed long to be queried.

rc ([ODSShort](#)) - returns
The position of the first bit set in the specified signed long.

ODFirstBit - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODFixedDivide

ODFixedDivide - Syntax

This function divides one fixed-point number by another fixed-point number.

```
#include <odmath.h>

ODFixed    a;
ODFixed    b;
ODFixed    rc;

rc = ODFixedDivide(a, b);
```

ODFixedDivide Parameter - a

a (ODFixed) - input
The dividend.

ODFixedDivide Parameter - b

b (ODFixed) - input
The divisor.

ODFixedDivide Return Value - rc

rc (ODFixed) - returns
The quotient.

ODFixedDivide - Parameters

a (ODFixed) - input
The dividend.

b (ODFixed) - input
The divisor.

rc (ODFixed) - returns
The quotient.

ODFixedDivide - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODFixedRound

ODFixedRound - Syntax

This macro rounds a fixed-point number to a signed short.

```
#include <odmath.h>

ODFixed    a;
ODSShort   rc;

rc = ODFixedRound(a);
```

ODFixedRound Parameter - a

a ([ODFixed](#)) - input

The fixed-point number to be rounded.

ODFixedRound Return Value - rc

rc ([ODSShort](#)) - returns

The signed short containing the rounded number.

ODFixedRound - Parameters

a ([ODFixed](#)) - input

The fixed-point number to be rounded.

rc ([ODSShort](#)) - returns
The signed short containing the rounded number.

ODFixedRound - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFixedMultiply

ODFixedMultiply - Syntax

This function multiplies two fixed-point numbers.

```
#include <odmath.h>

ODFixed    a;
ODFixed    b;
ODFixed    rc;

rc = ODFixedMultiply(a, b);
```

ODFixedMultiply Parameter - a

a ([ODFixed](#)) - input
The first fixed-point number to be multiplied.

ODFixedMultiply Parameter - b

b ([ODFixed](#)) - input
The second fixed-point number to be multiplied.

ODFixedMultiply Return Value - rc

rc ([ODFixed](#)) - returns
The product of the multiplication.

ODFixedMultiply - Parameters

a ([ODFixed](#)) - input
The first fixed-point number to be multiplied.

b ([ODFixed](#)) - input
The second fixed-point number to be multiplied.

rc ([ODFixed](#)) - returns
The product of the multiplication.

ODFixedMultiply - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFixedToFloat

ODFixedToFloat - Syntax

This macro converts a fixed-point number to a floating-point number.

```
#include <odmath.h>

ODFixed    a;
ODFloat    rc;

rc = ODFixedToFloat(a);
```

ODFixedToFloat Parameter - a

a ([ODFixed](#)) - input
The fixed-point number to be converted.

ODFixedToFloat Return Value - rc

rc ([ODFloat](#)) - returns
The converted floating-point number.

ODFixedToFloat - Parameters

a ([ODFixed](#)) - input
The fixed-point number to be converted.

rc ([ODFloat](#)) - returns
The converted floating-point number.

ODFixedToFloat - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFixedToFract

ODFixedToFract - Syntax

This macro converts a fixed-point number (16:16) to a fractional number (2:30).

```
#include <odmath.h>
```

```
ODFixed    a;  
ODFract    rc;
```

```
rc = ODFixedToFract(a);
```

ODFixedToFract Parameter - a

a ([ODFixed](#)) - input
The fixed-point number to be converted.

ODFixedToFract Return Value - rc

rc ([ODFract](#)) - returns
The converted fractional number.

ODFixedToFract - Parameters

a ([ODFixed](#)) - input
The fixed-point number to be converted.

rc ([ODFract](#)) - returns
The converted fractional number.

ODFixedToFract - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODFloatToFixed

ODFloatToFixed - Syntax

This macro converts a floating-point number to a fixed-point number (16:16).

```
#include <odmath.h>
```

```
ODFloat    a;  
ODFixed    rc;
```

```
rc = ODFloatToFixed(a);
```

ODFloatToFixed Parameter - a

a ([ODFloat](#)) - input
The floating-point number to be converted.

ODFloatToFixed Return Value - rc

rc ([ODFixed](#)) - returns
The converted fixed-point number.

ODFloatToFixed - Parameters

a ([ODFloat](#)) - input
The floating-point number to be converted.

rc ([ODFixed](#)) - returns
The converted fixed-point number.

ODFloatToFixed - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFloatToFract

ODFloatToFract - Syntax

This macro converts a floating-point number to a fractional number (2:30).

```
#include <odmath.h>

ODFloat    a;
ODFract    rc;

rc = ODFloatToFract(a);
```

ODFloatToFract Parameter - a

a ([ODFloat](#)) - input
The floating-point number to be converted.

ODFloatToFract Return Value - rc

rc ([ODFract](#)) - returns
The converted fractional number.

ODFloatToFract - Parameters

a ([ODFloat](#)) - input
The floating-point number to be converted.

rc ([ODFract](#)) - returns
The converted fractional number.

ODFloatToFract - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFractDivide

ODFractDivide - Syntax

This function divides one fractional number by another fractional number.

```
#include <odmath.h>
```

```
ODFract    a;  
ODFract    b;  
ODFract    rc;
```

```
rc = ODFractDivide(a, b);
```

ODFractDivide Parameter - a

a ([ODFract](#)) - input
The dividend.

ODFractDivide Parameter - b

b ([ODFract](#)) - input
The divisor.

ODFractDivide Return Value - rc

rc ([ODFract](#)) - returns
The quotient.

ODFractDivide - Parameters

a ([ODFract](#)) - input
The dividend.

b ([ODFract](#)) - input
The divisor.

rc ([ODFract](#)) - returns
The quotient.

ODFractDivide - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFractMultiply

ODFractMultiply - Syntax

This function multiplies two fractional numbers.

```
#include <odmath.h>

ODFract    a;
ODFract    b;
ODFract    rc;

rc = ODFractMultiply(a, b);
```

ODFractMultiply Parameter - a

a (ODFract) - input
The first fractional number to be multiplied.

ODFractMultiply Parameter - b

b (ODFract) - input
The second fractional number to be multiplied.

ODFractMultiply Return Value - rc

rc (ODFract) - returns
The product of the multiplication.

ODFractMultiply - Parameters

a (ODFract) - input
The first fractional number to be multiplied.

b (ODFract) - input
The second fractional number to be multiplied.

rc (ODFract) - returns
The product of the multiplication.

ODFractMultiply - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODFractSinCos

ODFractSinCos - Syntax

This function returns the sine and cosine of a fractional number.

```
#include <odmath.h>

ODFixed    angle;
ODFract    *cos;
ODFract    rc;

rc = ODFractSinCos(angle, cos);
```

ODFractSinCos Parameter - angle

angle ([ODFixed](#)) - input
The angle of the radius.

ODFractSinCos Parameter - cos

cos ([ODFract *](#)) - output
The cosine of the angle.

ODFractSinCos Return Value - rc

rc ([ODFract](#)) - returns
The sine of the angle.

ODFractSinCos - Parameters

angle ([ODFixed](#)) - input
The angle of the radius.

cos ([ODFract *](#)) - output
The cosine of the angle.

rc ([ODFract](#)) - returns
The sine of the angle.

ODFractSinCos - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFractToFixed

ODFractToFixed - Syntax

This macro converts a fractional number (2:30) to a fixed-point number (16:16).

```
#include <odmath.h>

ODFract    a;
ODFixed    rc;

rc = ODFractToFixed(a);
```

ODFractToFixed Parameter - a

a ([ODFract](#)) - input
The fractional number to be converted.

ODFractToFixed Return Value - rc

rc ([ODFixed](#)) - returns
The converted fixed-point number.

ODFractToFixed - Parameters

a ([ODFract](#)) - input
The fractional number to be converted.

rc ([ODFixed](#)) - returns
The converted fixed-point number.

ODFractToFixed - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODFractToFloat

ODFractToFloat - Syntax

This function converts a fractional number (2:30) to a floating-point number.

```
#include <odmath.h>
```

```
ODFract    a;  
ODFloat    rc;
```

```
rc = ODFractToFloat(a);
```

ODFractToFloat Parameter - a

a ([ODFract](#)) - input
The fractional number to be converted.

ODFractToFloat Return Value - rc

rc ([ODFloat](#)) - returns
The converted floating-point number.

ODFractToFloat - Parameters

a ([ODFract](#)) - input
The fractional number to be converted.

rc ([ODFloat](#)) - returns
The converted floating-point number.

ODFractToFloat - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

IntToODFixed

IntToODFixed - Syntax

This macro converts a signed short to a fixed-point number (16:16).

```
#include <odmath.h>
```

```
ODSShort    a;  
ODFixed     rc;
```

```
rc = IntToODFixed(a);
```

IntToODFixed Parameter - a

a ([ODSShort](#)) - input

The signed short to be converted.

IntToODFixed Return Value - rc

rc ([ODFixed](#)) - returns
The converted fixed-point number.

IntToODFixed - Parameters

a ([ODSShort](#)) - input
The signed short to be converted.

rc ([ODFixed](#)) - returns
The converted fixed-point number.

IntToODFixed - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideAdd

ODWideAdd - Syntax

This function adds two wide integers.

```
#include <odmath.h>

ODWide      *a;
const ODWide *b;
ODWide      *rc;

rc = ODWideAdd(a, b);
```

ODWideAdd Parameter - a

a ([ODWide *](#)) - input
The first wide integer to be added.

ODWideAdd Parameter - b

b ([const ODWide *](#)) - input
The second wide integer to be added.

ODWideAdd Return Value - rc

rc ([ODWide *](#)) - returns
The sum of the two wide integers.

ODWideAdd - Parameters

a ([ODWide *](#)) - input
The first wide integer to be added.

b ([const ODWide *](#)) - input
The second wide integer to be added.

rc ([ODWide *](#)) - returns
The sum of the two wide integers.

ODWideAdd - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideCompare

ODWideCompare - Syntax

This function compares two wide integers.

```
#include <odmath.h>

const ODWide      *a;
const ODWide      *b;
ODSShort          rc;

rc = ODWideCompare(a, b);
```

ODWideCompare Parameter - a

a (`const ODWide *`) - input
The first wide integer to be compared.

ODWideCompare Parameter - b

b (`const ODWide *`) - input
The second wide integer to be compared.

ODWideCompare Return Value - rc

rc (`ODSShort`) - returns
The result of the comparison. This parameter returns one of the following values:

1	$a > b$
0	$a = b$
-1	$a < b$

ODWideCompare - Parameters

a (`const ODWide *`) - input
The first wide integer to be compared.

b (`const ODWide *`) - input
The second wide integer to be compared.

rc (`ODSShort`) - returns
The result of the comparison. This parameter returns one of the following values:

1

	$a > b$
0	
	$a = b$
-1	
	$a < b$

ODWideCompare - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideDivide

ODWideDivide - Syntax

This function divides one wide integer by another wide integer.

```
#include <odmath.h>
```

```
const ODWide      *dividend;  
ODSLong           divisor;  
ODSLong           *remainder;  
ODSLong           rc;
```

```
rc = ODWideDivide(dividend, divisor, remainder);
```

ODWideDivide Parameter - dividend

dividend ([const ODWide *](#)) - input
The dividend.

ODWideDivide Parameter - divisor

divisor ([ODSLong](#)) - input
The divisor.

ODWideDivide Parameter - remainder

remainder ([ODSLong *](#)) - output
The remainder.

ODWideDivide Return Value - rc

rc ([ODSLong](#)) - returns
The quotient.

ODWideDivide - Parameters

dividend ([const ODSWide *](#)) - input
The dividend.

divisor ([ODSLong](#)) - input
The divisor.

remainder ([ODSLong *](#)) - output
The remainder.

rc ([ODSLong](#)) - returns
The quotient.

ODWideDivide - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideIsLong

ODWideIsLong - Syntax

This macro indicates whether a wide integer has less than 32 bits of significance.

```
#include <odmath.h>
```

```
ODWide      w;  
ODBoolean   rc;  
  
rc = ODWideIsLong(w);
```

ODWideIsLong Parameter - w

w ([ODWide](#)) - input
The wide integer to be checked.

ODWideIsLong Return Value - rc

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified wide integer has less than 32 bits of significance.

kODTrue	The specified wide integer has less than 32 bits of significance.
kODFalse	The specified wide integer has at least than 32 bits of significance.

ODWideIsLong - Parameters

w ([ODWide](#)) - input
The wide integer to be checked.

rc ([ODBoolean](#)) - returns
A flag indicating whether the specified wide integer has less than 32 bits of significance.

kODTrue	The specified wide integer has less than 32 bits of significance.
kODFalse	The specified wide integer has at least than 32 bits of significance.

ODWideIsLong - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideMultiply

ODWideMultiply - Syntax

This function multiplies two signed, long integers.

```
#include <odmath.h>

ODSLong    a;
ODSLong    b;
ODWide     *result;
ODWide     *rc;

rc = ODWideMultiply(a, b, result);
```

ODWideMultiply Parameter - a

a (ODSLong) - input
The first integer to be multiplied.

ODWideMultiply Parameter - b

b (ODSLong) - input
The second integer to be multiplied.

ODWideMultiply Parameter - result

result (ODWide *) - output
A pointer to the product of the multiplication.

ODWideMultiply Return Value - rc

rc (ODWide *) - returns
A pointer to the product of the multiplication.

ODWideMultiply - Parameters

a ([ODSLong](#)) - input
The first integer to be multiplied.

b ([ODSLong](#)) - input
The second integer to be multiplied.

result ([ODWide *](#)) - output
A pointer to the product of the multiplication.

rc ([ODWide *](#)) - returns
A pointer to the product of the multiplication.

ODWideMultiply - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideNegate

ODWideNegate - Syntax

This function negates a wide integer.

```
#include <odmath.h>
```

```
ODWide      *w;  
ODWide      *rc;
```

```
rc = ODWideNegate(w);
```

ODWideNegate Parameter - w

w ([ODWide *](#)) - in/out
On input, this parameter is the wide integer to be negated. On output, the 2's complement of the wide integer is returned.

ODWideNegate Return Value - rc

rc ([ODWide *](#)) - returns
The 2's complement of the specified wide integer.

ODWideNegate - Parameters

w ([ODWide *](#)) - in/out
On input, this parameter is the wide integer to be negated. On output, the 2's complement of the wide integer is returned.

rc ([ODWide *](#)) - returns
The 2's complement of the specified wide integer.

ODWideNegate - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideShift

ODWideShift - Syntax

This function shifts a wide integer by the specified number of bits.

```
#include <odmath.h>

ODWide      *w;
ODSShort    bits;
ODWide      *rc;

rc = ODWideShift(w, bits);
```

ODWideShift Parameter - w

w ([ODWide *](#)) - in/out
The wide integer to be shifted.

ODWideShift Parameter - bits

bits ([ODSShort](#)) - input

The number of bits to shift the wide integer. This parameter can be set to one of the following values:

>0	Shifts to the right.
<0	Shifts to the left.
0	Does not shift.

ODWideShift Return Value - rc

rc ([ODWide *](#)) - returns

The wide integer that resulted from the shift.

ODWideShift - Parameters

w ([ODWide *](#)) - in/out

The wide integer to be shifted.

bits ([ODSShort](#)) - input

The number of bits to shift the wide integer. This parameter can be set to one of the following values:

>0	Shifts to the right.
<0	Shifts to the left.
0	Does not shift.

rc ([ODWide *](#)) - returns

The wide integer that resulted from the shift.

ODWideShift - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideSquareRoot

ODWideSquareRoot - Syntax

This function returns the approximate square root of a wide integer.

```
#include <odmath.h>

const ODWide      *src;
ODULong           rc;

rc = ODWideSquareRoot(src);
```

ODWideSquareRoot Parameter - src

src ([const ODWide *](#)) - input

A pointer to the wide integer whose square root is being requested. This must be a positive integer.

ODWideSquareRoot Return Value - rc

rc ([ODULong](#)) - returns

The approximate square root of the specified wide integer.

ODWideSquareRoot - Parameters

src ([const ODWide *](#)) - input

A pointer to the wide integer whose square root is being requested. This must be a positive integer.

rc ([ODULong](#)) - returns

The approximate square root of the specified wide integer.

ODWideSquareRoot - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODWideSubtract

ODWideSubtract - Syntax

This function subtracts two wide integers.

```
#include <odmath.h>

ODWide      *a;
const ODWide *b;
ODWide      *rc;

rc = ODWideSubtract(a, b);
```

ODWideSubtract Parameter - a

a (ODWide *) - in/out

On input, this parameter is the wide integer to be subtracted from. On output, the difference between *a* and *b* is returned.

ODWideSubtract Parameter - b

b (const ODWide *) - input

The wide integer to be subtracted.

ODWideSubtract Return Value - rc

rc (ODWide *) - returns

The difference between *a* and *b*.

ODWideSubtract - Parameters

a (ODWide *) - in/out

On input, this parameter is the wide integer to be subtracted from. On output, the difference between *a* and *b* is returned.

b (const ODWide *) - input

The wide integer to be subtracted.

rc (ODWide *) - returns

The difference between *a* and *b*.

ODWideSubtract - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

OSA Descriptor Utilities

These functions are used by part handlers that want to use Open Scripting Architecture (OSA). They allow conversion between [AEDesc](#) structures understood by OS/2 and [ODDescType](#) structures understood by OpenDoc.

The following lists the OSA descriptor utilities in alphabetic order.

- [ODDescToAEDesc](#)
- [AEDescToODDesc](#)

ODDescToAEDesc

ODDescToAEDesc - Syntax

This function copies data from the specified [ODDesc](#) object to an [AEDesc](#) structure.

```
#include <oddesutl.h>

ODDesc      *odDesc;
AEDesc      *aeDesc;
ODError     rv;

rv = ODDescToAEDesc(odDesc, aeDesc);
```

ODDescToAEDesc Parameter - odDesc

odDesc (ODDesc *) - input
The OpenDoc descriptor record object to be converted.

ODDescToAEDesc Parameter - aeDesc

aeDesc ([AEDesc *](#)) - output

The converted descriptor record structure. When it is finished using this value, this caller must dispose of it.

ODDescToAEDesc Return Value - rv

rv ([ODError](#)) - returns

The error code, or 0 if the operation was successful.

ODDescToAEDesc - Parameters

odDesc ([ODDesc *](#)) - input

The OpenDoc descriptor record object to be converted.

aeDesc ([AEDesc *](#)) - output

The converted descriptor record structure. When it is finished using this value, this caller must dispose of it.

rv ([ODError](#)) - returns

The error code, or 0 if the operation was successful.

ODDescToAEDesc - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

AEDescToODDesc

AEDescToODDesc - Syntax

This function copies the data from the specified [AEDesc](#) structure into the specified [ODDesc](#) object.

```
#include <oddesutl.h>
```

```
AEDesc      *aeDesc;  
ODDesc      *odDesc;  
ODError      rv;
```

```
rv = AEDescToODDesc(aeDesc, odDesc);
```

AEDescToODDesc Parameter - aeDesc

aeDesc ([AEDesc *](#)) - input

The descriptor record structure to be converted. When it is finished using this value, the caller must dispose of it.

AEDescToODDesc Parameter - odDesc

odDesc ([ODDesc *](#)) - output

The converted OpenDoc descriptor record object. This parameter must contain a valid instantiation, but can be empty.

AEDescToODDesc Return Value - rv

rv ([ODError](#)) - returns

The error code, or 0 if the operation was successful.

AEDescToODDesc - Parameters

aeDesc ([AEDesc *](#)) - input

The descriptor record structure to be converted. When it is finished using this value, the caller must dispose of it.

odDesc ([ODDesc *](#)) - output

The converted OpenDoc descriptor record object. This parameter must contain a valid instantiation, but can be empty.

rv ([ODError](#)) - returns

The error code, or 0 if the operation was successful.

AEDescToODDesc - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

Persistent Objects

This section contains help functions for use with persistent objects. At initialization time (InitXXXX or InitXXXXFromStorage), the client should call the appropriate initialization functions. At clone-into time, the client should call the corresponding clone functions. At externalization time,

the client should call the appropriate update functions.

The following lists the persistent object functions in alphabetic order.

- [CloneDateInfo](#)
- [CloneModificationInfo](#)
- [InitDateInfo](#)
- [InitModificationInfo](#)
- [UpdateDateInfo](#)
- [UpdateModificationInfo](#)

CloneDateInfo

CloneDateInfo - Syntax

This function clones date information in a storage unit at clone-into time.

```
#include <poutils.h>

Environment      *ev;
ODStorageUnit    *fromSU;
ODStorageUnit    *toSU;

CloneDateInfo(ev, fromSU, toSU);
```

CloneDateInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

CloneDateInfo Parameter - fromSU

fromSU ([ODStorageUnit *](#)) - input
The storage unit from which information is to be cloned.

CloneDateInfo Parameter - toSU

toSU ([ODStorageUnit *](#)) - input
The storage unit to which information is to be cloned.

CloneDateInfo - Return Value

None.

CloneDateInfo - Parameters

ev ([Environment](#) *) - input
The SOM environment.

fromSU (ODStorageUnit *) - input
The storage unit from which information is to be cloned.

toSU (ODStorageUnit *) - input
The storage unit to which information is to be cloned.

None.

CloneDateInfo - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

CloneModificationInfo

CloneModificationInfo - Syntax

This function clones modification information in a storage unit at clone-into time.

```
#include <poutils.h>

Environment      *ev;
ODStorageUnit    *fromSU;
ODStorageUnit    *toSU;

CloneModificationInfo(ev, fromSU, toSU);
```

CloneModificationInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

CloneModificationInfo Parameter - fromSU

fromSU ([ODStorageUnit *](#)) - input
The storage unit from which information is to be cloned.

CloneModificationInfo Parameter - toSU

toSU ([ODStorageUnit *](#)) - input
The storage unit to which information is to be cloned.

CloneModificationInfo - Return Value

None.

CloneModificationInfo - Parameters

ev ([Environment *](#)) - input
The SOM environment.

fromSU ([ODStorageUnit *](#)) - input
The storage unit from which information is to be cloned.

toSU ([ODStorageUnit *](#)) - input
The storage unit to which information is to be cloned.

None.

CloneModificationInfo - Topics

Select an item:

InitDateInfo

InitDateInfo - Syntax

This function initializes date information in a storage unit at initialization time.

```
#include <poutils.h>

Environment      *ev;
ODStorageUnit    *su;

InitDateInfo(ev, su);
```

InitDateInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

InitDateInfo Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit into which data is to be added.

InitDateInfo - Return Value

None.

InitDateInfo - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit into which data is to be added.

None.

InitDateInfo - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

InitModificationInfo

InitModificationInfo - Syntax

This function initializes modification information in a storage unit at initialization time.

```
#include <poutils.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;
```

```
InitModificationInfo(ev, su);
```

InitModificationInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

InitModificationInfo Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to which modification information is to be added.

InitModificationInfo - Return Value

None.

InitModificationInfo - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to which modification information is to be added.

None.

InitModificationInfo - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

UpdateDateInfo

UpdateDateInfo - Syntax

This function updates date information in a storage unit at externalization time.

```
#include <poutils.h>

Environment      *ev;
ODStorageUnit    *su;

UpdateDateInfo(ev, su);
```

UpdateDateInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

UpdateDateInfo Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to externalize to.

UpdateDateInfo - Return Value

None.

UpdateDateInfo - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to externalize to.

None.

UpdateDateInfo - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

UpdateModificationInfo

UpdateModificationInfo - Syntax

This function updates modification information in a storage unit at externalization time.

```
#include <poutils.h>

Environment      *ev;
ODStorageUnit    *su;

UpdateModificationInfo(ev, su);
```

UpdateModificationInfo Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

UpdateModificationInfo Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to externalize to.

UpdateModificationInfo - Return Value

None.

UpdateModificationInfo - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to externalize to.

None.

UpdateModificationInfo - Topics

Select an item:

Registration

A registry is a table that maps data types to executable code. These tables represent the preferences of the user for particular part editors, scripting systems, and translators.

Several different registry objects are supported, including parts, scripting systems, and translations. Part registries allow the document shell to map parts to part editors according to their part kind. Scripting system registries determine what scripting system is used to edit and interpret a script. Translation registries regulate which translation module is used for translating data of a given part kind into data of a different part kind.

Name spaces are used to uniquely identify part kinds, translation module kinds, scripting systems, and object extensions for code-binding purposes. A name space associates an ISO string (for example, a part kind) with a pointer to the code (for example, the part editor for the part kind).

The following lists the registration functions in alphabetic order.

- [ODDeregisterPartHandler](#)
- [ODDeregisterPartHandlerClass](#)
- [ODQueryPartHandlerInfo](#)
- [ODQueryPartHandlerList](#)
- [ODQueryPartKindInfo](#)
- [ODQueryPartKindList](#)
- [ODQueryPreferredPartHandler](#)
- [ODQueryPreferredPartHandlerForCategory](#)
- [ODQueryPreferredPartHandlerForFileExt](#)
- [ODQueryPreferredPartHandlerForFileType](#)
- [ODRegisterPartHandlerClass](#)
- [ODSetPreferredPartHandler](#)
- [ODSetPreferredPartHandlerForCategory](#)
- [ODSetPreferredPartHandlerForFileExt](#)
- [ODSetPreferredPartHandlerForFileType](#)

ODDeregisterPartHandler

ODDeregisterPartHandler - Syntax

This function deregisters a part handler with the OpenDoc registry and the OpenDoc SOM interface repository using the specified part handler name.

```
#include <odregapi.h>

ISOString    isoPartHandlerName;
APIRET      rc; /* Return codes. */

rc = ODDeregisterPartHandler(isoPartHandlerName);
```

ODDeregisterPartHandler Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - input
The name of the part handle to be deregistered.

ODDeregisterPartHandler Return Value - rc

rc ([APIRET](#)) - returns
Return codes.

NO_ERROR

ERROR_COULD_NOT_FIND_PART_HANDLER

ERROR_COUNUD_NOT_FIND_CLASS

ERROR_COUNUD_NOT_FIND_CLASS_IN_SOM_IR

ODDeregisterPartHandler - Parameters

isoPartHandlerName ([ISOString](#)) - input
The name of the part handle to be deregistered.

rc ([APIRET](#)) - returns
Return codes.

NO_ERROR

ERROR_COULD_NOT_FIND_PART_HANDLER

ERROR_COUNUD_NOT_FIND_CLASS

ERROR_COUNUD_NOT_FIND_CLASS_IN_SOM_IR

ODDeregisterPartHandler - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODDeregisterPartHandlerClass

ODDeregisterPartHandlerClass - Syntax

This function deregisters a part handler with the OpenDoc registry and the OpenDoc SOM interface repository using the specified class name.

```
#include <odregapi.h>

PSZ      pszClassName;
APIRET   rc;          /* Return codes. */

rc = ODDeregisterPartHandlerClass(pszClassName);
```

ODDeregisterPartHandlerClass Parameter - pszClassName

pszClassName ([PSZ](#)) - input
The name of the class of the part handler to be deregistered.

ODDeregisterPartHandlerClass Return Value - rc

rc ([APIRET](#)) - returns
Return codes.

- NO_ERROR
- ERROR_COULD_NOT_FIND_PART_HANDLER
- ERROR_COUNUD_NOT_FIND_CLASS
- ERROR_COUNUD_NOT_FIND_CLASS_IN_SOM_IR

ODDeregisterPartHandlerClass - Parameters

pszClassName ([PSZ](#)) - input
The name of the class of the part handler to be deregistered.

rc ([APIRET](#)) - returns
Return codes.

- NO_ERROR
- ERROR_COULD_NOT_FIND_PART_HANDLER
- ERROR_COUNUD_NOT_FIND_CLASS
- ERROR_COUNUD_NOT_FIND_CLASS_IN_SOM_IR

ODDeregisterPartHandlerClass - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPartHandlerInfo

ODQueryPartHandlerInfo - Syntax

This function returns information about the specified part handler.

```
#include <odregapi.h>

ISOString      isoPartHandlerName;
PartKindQueryInfo *partKindQueryInfo;
APIRET         rc; /* Return code. */

rc = ODQueryPartHandlerInfo(isoPartHandlerName,
                             partKindQueryInfo);
```

ODQueryPartHandlerInfo Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - input

The part handler for which information is to be returned.

ODQueryPartHandlerInfo Parameter - partKindQueryInfo

partKindQueryInfo ([PartKindQueryInfo *](#)) - output

Information about the specified part handler.

ODQueryPartHandlerInfo Return Value - rc

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

=====

=====

=====

ODQueryPartHandlerList Parameter - isoPartKind

isoPartKind ([ISOString](#)) - input

The part kind that the part handlers can edit. This parameter is optional and can be set to KODNULL.

ODQueryPartHandlerList Parameter - pszCategory

pszCategory ([PSZ](#)) - input

The category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to KODNULL.

ODQueryPartHandlerList Parameter - pszBuffer

pszBuffer ([PSZ](#)) - output

A list of part handlers. This buffer contains the names separated by commas.

ODQueryPartHandlerList Parameter - pulBufferSize

pulBufferSize ([PULONG](#)) - in/out

On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the list of part handlers in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

ODQueryPartHandlerList Return Value - rc

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPartHandlerList - Parameters

isoPartKind ([ISOString](#)) - input

The part kind that the part handlers can edit. This parameter is optional and can be set to KODNULL.

pszCategory ([PSZ](#)) - input

The category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to kODNULL.

pszBuffer ([PSZ](#)) - output

A list of part handlers. This buffer contains the names separated by commas.

pulBufferSize ([PULONG](#)) - in/out

On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the list of part handlers in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPartHandlerList - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPartKindInfo

ODQueryPartKindInfo - Syntax

This function returns information about the specified part kind.

```
#include <odregapi.h>
```

```
ISOString          isoPartHandlerName;  
ISOString          isoPartKindName;  
PartKindQueryInfo *partKindQueryInfo;  
APIRET            rc;          /* Return code. */
```

```
rc = ODQueryPartKindInfo(isoPartHandlerName,  
                          isoPartKindName, partKindQueryInfo);
```

ODQueryPartKindInfo Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - input

The part handler name for which information is to be returned.

ODQueryPartKindInfo Parameter - isoPartKindName

isoPartKindName ([ISOString](#)) - input
The part kind for which information is to be returned.

ODQueryPartKindInfo Parameter - partKindQueryInfo

partKindQueryInfo ([PartKindQueryInfo *](#)) - output
A structure containing information about the part kind.

ODQueryPartKindInfo Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ERROR_COULD_NOT_FIND_PART_HANDLER

ODQueryPartKindInfo - Parameters

isoPartHandlerName ([ISOString](#)) - input
The part handler name for which information is to be returned.

isoPartKindName ([ISOString](#)) - input
The part kind for which information is to be returned.

partKindQueryInfo ([PartKindQueryInfo *](#)) - output
A structure containing information about the part kind.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ERROR_COULD_NOT_FIND_PART_HANDLER

ODQueryPartKindInfo - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPartKindList

ODQueryPartKindList - Syntax

This function returns a list of part kinds that meets the given criteria. If the part handler and category are not specified, a list of all part-registered part kinds is returned.

```
#include <odregapi.h>

ISOString    isoPartHandlerName;
PSZ          pszCategory;
PSZ          *pszBuffer;
PULONG       pulBufferSize;
APIRET       rc;                /* Return code. */

rc = ODQueryPartKindList(isoPartHandlerName,
                        pszCategory, pszBuffer, pulBufferSize);
```

ODQueryPartKindList Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - input

The part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

ODQueryPartKindList Parameter - pszCategory

pszCategory ([PSZ](#)) - input

The category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

ODQueryPartKindList Parameter - pszBuffer

pszBuffer ([PSZ *](#)) - output

A list of part kinds. This buffer contains the part kind names separated by commas.

When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

ODQueryPartKindList Parameter - pulBufferSize

pulBufferSize ([PULONG](#)) - in/out

On input, this parameter is the size, in bytes of the specified buffer. On output, the actual size of the list of part kinds placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

ODQueryPartKindList Return Value - rc

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPartKindList - Parameters

isoPartHandlerName ([ISOString](#)) - input

The part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

pszCategory ([PSZ](#)) - input

The category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

pszBuffer ([PSZ *](#)) - output

A list of part kinds. This buffer contains the part kind names separated by commas.

When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

pulBufferSize ([PULONG](#)) - in/out

On input, this parameter is the size, in bytes of the specified buffer. On output, the actual size of the list of part kinds placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPartKindList - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPreferredPartHandler

ODQueryPreferredPartHandler - Syntax

This function returns the preferred part-handler name for the specified part kind.

```
#include <odregapi.h>

ISOString    isoPartKindName;
ISOStr       isoPartHandlerName;
PULONG       pulBufferSize;
APIRET       rc;                /* Return code. */

rc = ODQueryPreferredPartHandler(isoPartKindName,
                                isoPartHandlerName, pulBufferSize);
```

ODQueryPreferredPartHandler Parameter - isoPartKindName

isoPartKindName (ISOString) - input
The part kind name.

ODQueryPreferredPartHandler Parameter - isoPartHandlerName

isoPartHandlerName (ISOStr) - output
A buffer in which the preferred part handler for the specified part kind is to be returned.

ODQueryPreferredPartHandler Parameter - pulBufferSize

pulBufferSize (PULONG) - in/out
On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer size is too small, the size of the buffer required to hold the string is returned.

ODQueryPreferredPartHandler Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ERROR_COULD_NOT_FIND_PART_KIND

ODQueryPreferredPartHandler - Parameters

isoPartKindName ([ISOStr](#)) - input
The part kind name.

isoPartHandlerName ([ISOStr](#)) - output
A buffer in which the preferred part handler for the specified part kind is to be returned.

pulBufferSize ([PULONG](#)) - in/out
On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer size is too small, the size of the buffer required to hold the string is returned.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ERROR_COULD_NOT_FIND_PART_KIND

ODQueryPreferredPartHandler - Topics

Select an item:

- [Syntax](#)
- [Parameters](#)
- [Returns](#)

ODQueryPreferredPartHandlerForCategory

ODQueryPreferredPartHandlerForCategory - Syntax

This function returns information about the preferred part-handler using the specified category.

```
#include <odregapi.h>

ISOStr      isoCategory;
ISOStr      isoPartHandlerName;
PULONG      pulBufferSize;
```

```
APIRET rc; /* Return code. */

rc = ODQueryPreferredPartHandlerForCategory(
    isoCategory, isoPartHandlerName, pulBufferSize);
```

ODQueryPreferredPartHandlerForCategory Parameter - isoCategory

isoCategory (ISOString) - input
The category of the part.

ODQueryPreferredPartHandlerForCategory Parameter - isoPartHandlerName

isoPartHandlerName (ISOString) - output
A buffer in which the preferred part handler for the specified part category is to be returned.

ODQueryPreferredPartHandlerForCategory Parameter - pulBufferSize

pulBufferSize (PULONG) - input
On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

ODQueryPreferredPartHandlerForCategory Return Value - rc

rc (APIRET) - returns
Return code.

ODQueryPreferredPartHandlerForCategory - Parameters

isoCategory (ISOString) - input
The category of the part.

isoPartHandlerName (ISOString) - output
A buffer in which the preferred part handler for the specified part category is to be returned.

pulBufferSize (PULONG) - input
On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

rc (APIRET) - returns

Return code.

ODQueryPreferredPartHandlerForCategory - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPreferredPartHandlerForFileExt

ODQueryPreferredPartHandlerForFileExt - Syntax

This function returns information about the preferred part-handler using the specified file extension.

```
#include <odregapi.h>

ISOString    isoFileExt;
ISOString    isoPartHandlerName;
PULONG      pulBufferSize;
APIRET      rc;          /* Return code. */

rc = ODQueryPreferredPartHandlerForFileExt(
    isoFileExt, isoPartHandlerName, pulBufferSize);
```

ODQueryPreferredPartHandlerForFileExt Parameter - isoFileExt

isoFileExt ([ISOString](#)) - input
The file extension of the part.

ODQueryPreferredPartHandlerForFileExt Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - output
A buffer in which the preferred part handler for the specified file extension is to be returned.

ODQueryPreferredPartHandlerForFileExt Parameter - pulBufferSize

pulBufferSize ([PULONG](#)) - input

On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

ODQueryPreferredPartHandlerForFileExt Return Value - rc

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPreferredPartHandlerForFileExt - Parameters

isoFileExt ([ISOString](#)) - input

The file extension of the part.

isoPartHandlerName ([ISOString](#)) - output

A buffer in which the preferred part handler for the specified file extension is to be returned.

pulBufferSize ([PULONG](#)) - input

On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

rc ([APIRET](#)) - returns

Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPreferredPartHandlerForFileExt - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODQueryPreferredPartHandlerForFileType

ODQueryPreferredPartHandlerForFileType - Syntax

This method returns the preferred part-handler name for the specified file type.

```
#include <odregapi.h>

ISOString    partFileType;
ISOString    isoPartHandlerName;
PULONG      pulBufferSize;
APIRET      rc;          /* Return code. */

rc = ODQueryPreferredPartHandlerForFileType(
    partFileType, isoPartHandlerName, pulBufferSize);
```

ODQueryPreferredPartHandlerForFileType Parameter - partFileType

partFileType ([ISOString](#)) - input
The file type of the part.

ODQueryPreferredPartHandlerForFileType Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - output
A buffer in which the preferred part handler for the specified file type is to be returned.

ODQueryPreferredPartHandlerForFileType Parameter - pulBufferSize

pulBufferSize ([PULONG](#)) - input
On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

ODQueryPreferredPartHandlerForFileType Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPreferredPartHandlerForFileType - Parameters

partFileType ([ISOString](#)) - input
The file type of the part.

isoPartHandlerName ([ISOString](#)) - output
A buffer in which the preferred part handler for the specified file type is to be returned.

pulBufferSize ([PULONG](#)) - input
On input, this parameter is the size, in bytes, of the buffer. On output, the actual size of the string placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

errPR_BUFFER_TOO_SMALL

ODQueryPreferredPartHandlerForFileType - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODRegisterPartHandlerClass

ODRegisterPartHandlerClass - Syntax

This function registers a part handler with the OpenDoc registry.

```
#include <odregapi.h>
```

```
PSZ      pszClassName;  
PSZ      pszDllName;  
BOOL     cTemplate;  
long     reserved;  
APIRET   rc;           /* Return codes. */
```

```
rc = ODRegisterPartHandlerClass(pszClassName,  
                                pszDllName, cTemplate, reserved);
```

ODRegisterPartHandlerClass Parameter - pszClassName

pszClassName ([PSZ](#)) - input
The name of the SOM class to be registered.

ODRegisterPartHandlerClass Parameter - pszDllName

pszDllName ([PSZ](#)) - input
The qualified or unqualified DLL name of the class.

ODRegisterPartHandlerClass Parameter - cTemplate

cTemplate ([BOOL](#)) - input
A flag indicating whether a template should be created.

TRUE	A template should be created.
FALSE	A template should not be created.

ODRegisterPartHandlerClass Parameter - reserved

reserved (long) - input
Reserved value, should be 0.

ODRegisterPartHandlerClass Return Value - rc

rc ([APIRET](#)) - returns
Return codes.

NO_ERROR
errPR_INVALID_CLASSNAME
errPR_COULD_NOT_LOAD_CLASS
errPR_NOT_PART_HANDLER_CLASS
errPR_PART_HANDLER_INFO_REPLACED
errPR_PART_HANDLER_NOT_FOUND
errPR_PART_KIND_NOT_FOUND
errPR_BUFFER_TOO_SMALL

ODRegisterPartHandlerClass - Parameters

pszClassName ([PSZ](#)) - input

The name of the SOM class to be registered.

pszDllName ([PSZ](#)) - input

The qualified or unqualified DLL name of the class.

cTemplate ([BOOL](#)) - input

A flag indicating whether a template should be created.

TRUE

A template should be created.

FALSE

A template should not be created.

reserved (long) - input

Reserved value, should be 0.

rc ([APIRET](#)) - returns

Return codes.

NO_ERROR

errPR_INVALID_CLASSNAME

errPR_COULD_NOT_LOAD_CLASS

errPR_NOT_PART_HANDLER_CLASS

errPR_PART_HANDLER_INFO_REPLACED

errPR_PART_HANDLER_NOT_FOUND

errPR_PART_KIND_NOT_FOUND

errPR_BUFFER_TOO_SMALL

ODRegisterPartHandlerClass - Remarks

This function creates an instance of the specified class using the specified DLL. The successful completion of this function depends on the information that is provided by the part-handler class. If the handler has already been registered, this function replaces the handler information.

ODRegisterPartHandlerClass - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

ODSetPreferredPartHandler

ODSetPreferredPartHandler - Syntax

This function sets the preferred part-handler name.

```
#include <odregapi.h>

ISOString    isoPartKindName;
ISOString    isoPartHandleName;
APIRET      rc;          /* Return codes. */

rc = ODSetPreferredPartHandler(isoPartKindName,
                               isoPartHandleName);
```

ODSetPreferredPartHandler Parameter - isoPartKindName

isoPartKindName ([ISOString](#)) - input
The part kind name.

ODSetPreferredPartHandler Parameter - isoPartHandleName

isoPartHandleName ([ISOString](#)) - input
The part handler name.

ODSetPreferredPartHandler Return Value - rc

rc ([APIRET](#)) - returns
Return codes.

- NO_ERROR
- ERROR_COULD_NOT_FIND_PART_KIND

ODSetPreferredPartHandler - Parameters

isoPartKindName ([ISOString](#)) - input
The part kind name.

isoPartHandlerName ([ISOString](#)) - input
The part handler name.

rc ([APIRET](#)) - returns
Return codes.

NO_ERROR

ERROR_COULD_NOT_FIND_PART_KIND

ODSetPreferredPartHandler - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODSetPreferredPartHandlerForCategory

ODSetPreferredPartHandlerForCategory - Syntax

This function sets the preferred part-handler to the specified name using the specified category.

```
#include <odregapi.h>

ISOString    isoCategory;
ISOString    isoPartHandlerName;
APIRET       rc;          /* Return code. */

rc = ODSetPreferredPartHandlerForCategory(
    isoCategory, isoPartHandlerName);
```

ODSetPreferredPartHandlerForCategory Parameter - isoCategory

isoCategory ([ISOString](#)) - input
The category of the part.

ODSetPreferredPartHandlerForCategory Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

ODSetPreferredPartHandlerForCategory Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ERROR_COULD_NOT_FIND_PART_KIND

ODSetPreferredPartHandlerForCategory - Parameters

isoCategory ([ISOString](#)) - input
The category of the part.

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ERROR_COULD_NOT_FIND_PART_KIND

ODSetPreferredPartHandlerForCategory - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetPreferredPartHandlerForFileExt

ODSetPreferredPartHandlerForFileExt - Syntax

This function sets the preferred part-handler to the specified name using the specified file extension.

```
#include <odregapi.h>
```

```
ISOString    isoFileExt;  
ISOString    isoPartHandlerName;  
APIRET      rc;          /* Return code. */
```

```
rc = ODSetPreferredPartHandlerForFileExt(  
    isoFileExt, isoPartHandlerName);
```

ODSetPreferredPartHandlerForFileExt Parameter - isoFileExt

isoFileExt ([ISOString](#)) - input
The file extension of the part.

ODSetPreferredPartHandlerForFileExt Parameter - isoPartHandlerName

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

ODSetPreferredPartHandlerForFileExt Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ODSetPreferredPartHandlerForFileExt - Parameters

isoFileExt ([ISOString](#)) - input
The file extension of the part.

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ODSetPreferredPartHandlerForFileExt - Topics

Select an item:

ODSetPreferredPartHandlerForFileType

ODSetPreferredPartHandlerForFileType - Syntax

This function sets the preferred part-handler to the specified name using the specified file type.

```
#include <odregapi.h>

ISOString    isoFileType;
ISOString    isoPartHandlerName;
APIRET      rc;          /* Return code. */

rc = ODSetPreferredPartHandlerForFileType(
    isoFileType, isoPartHandlerName);
```

ODSetPreferredPartHandlerForFileType Parameter - isoFileType

isoFileType ([ISOString](#)) - input
The file type name.

ODSetPreferredPartHandlerForFileType Parameter - isoPartHandle

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

ODSetPreferredPartHandlerForFileType Return Value - rc

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ODSetPreferredPartHandlerForFileType - Parameters

isoFileType ([ISOString](#)) - input
The file type name.

isoPartHandlerName ([ISOString](#)) - output
The part handler name.

rc ([APIRET](#)) - returns
Return code.

NO_ERROR

ODSetPreferredPartHandlerForFileType - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

Standard-Typed Values

The functions in this section are used to read standard-type values to and write standard-type values from storage units.

The following lists the functions and macros for standard-typed values in alphabetic order.

- [ODGetBooleanProp](#)
- [ODGetISOStrProp](#)
- [ODGetTextProp](#)
- [ODGetMatrixProp](#)
- [ODGetPointProp](#)
- [ODGetPolygonProp](#)
- [ODGetRectProp](#)
- [ODGetSLongProp](#)
- [ODGetSShortProp](#)
- [ODGetStrongSUNRefProp](#)
- [ODGetTime_TProp](#)
- [ODGetTypeListProp](#)
- [ODGetULongProp](#)
- [ODGetUShortProp](#)
- [ODGetWeakSUNRefProp](#)
- [ODSetBooleanProp](#)
- [ODSetISOStrProp](#)
- [ODSetTextProp](#)
- [ODSetMatrixProp](#)
- [ODSetPointProp](#)
- [ODSetPolygonProp](#)
- [ODSetRectProp](#)
- [ODSetSLongProp](#)
- [ODSetSShortProp](#)
- [ODSetStrongSUNRefProp](#)
- [ODSetTime_TProp](#)
- [ODSetTypeListProp](#)
- [ODSetULongProp](#)
- [ODSetUShortProp](#)
- [ODSetWeakSUNRefProp](#)
- [ODSUAddPropValue](#)
- [ODSUForceFocus](#)
- [ODSUExistsThenFocus](#)
- [ODSUNRemoveProperty](#)

ODGetBooleanProp

ODGetBooleanProp - Syntax

This function returns the boolean value from the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODBoolean        rv;

rv = ODGetBooleanProp(ev, su, prop, val);
```

ODGetBooleanProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetBooleanProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetBooleanProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetBooleanProp Parameter - val

val ([ODValueType](#)) - input

The value to be queried.

ODGetBooleanProp Return Value - rv

rv ([ODBoolean](#)) - returns

The boolean value from the storage unit, property and value.

kODTrue

The operation was successful.

kODFalse

The operation was unsuccessful.

ODGetBooleanProp - Parameters

ev ([Environment *](#)) - input

The SOM environment.

su ([ODStorageUnit *](#)) - input

The storage unit to be queried.

prop ([ODPropertyName](#)) - input

The property to be queried.

val ([ODValueType](#)) - input

The value to be queried.

rv ([ODBoolean](#)) - returns

The boolean value from the storage unit, property and value.

kODTrue

The operation was successful.

kODFalse

The operation was unsuccessful.

ODGetBooleanProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODGetISOStrProp

ODGetISOStrProp - Syntax

This function returns the ISO string from the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODISOStr         value;  
ODULong          *size;  
ODISOStr         rv;
```

```
rv = ODGetISOStrProp(ev, su, prop, val, value,  
                    size);
```

ODGetISOStrProp Parameter - ev

ev ([Environment *](#)) - input
The SOM Environment.

ODGetISOStrProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetISOStrProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetISOStrProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetISOStrProp Parameter - value

value ([ODISOStr](#)) - in/out

The ISO string value from the specified storage unit, property and value.

ODGetISOStrProp Parameter - size

size ([ODULong *](#)) - in/out

On input, the length of the longest ISO string that can be written to the *value* parameter, including the null-terminator. On output, the size is updated with the actual number of bytes read.

ODGetISOStrProp Return Value - rv

rv ([ODISOStr](#)) - returns

The ISO string value of the specified storage unit, property and value.

ODGetISOStrProp - Parameters

ev ([Environment *](#)) - input

The SOM Environment.

su ([ODStorageUnit *](#)) - input

The storage unit to be queried.

prop ([ODPropertyName](#)) - input

The property to be queried.

val ([ODValueType](#)) - input

The value to be queried.

value ([ODISOStr](#)) - in/out

The ISO string value from the specified storage unit, property and value.

size ([ODULong *](#)) - in/out

On input, the length of the longest ISO string that can be written to the *value* parameter, including the null-terminator. On output, the size is updated with the actual number of bytes read.

rv ([ODISOStr](#)) - returns

The ISO string value of the specified storage unit, property and value.

ODGetISOStrProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODGetITextProp

ODGetITextProp - Syntax

This function returns an IText structure from the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODIText          *name;
ODIText          *rv;

rv = ODGetITextProp(ev, su, prop, val, name);
```

ODGetITextProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetITextProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetITextProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetITextProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetITextProp Parameter - name

name ([ODIText *](#)) - in/out

The IText structure of the specified storage unit, property and value. If this value is KODNULL, the existing buffer of text in IText is allocated and filled; otherwise, the buffer of text is disposed of and a new one is allocated and filled.

ODGetITextProp Return Value - rv

rv ([ODIText *](#)) - returns

The IText structure of the specified storage unit, property and value.

ODGetITextProp - Parameters

ev ([Environment *](#)) - input

The SOM environment.

su ([ODStorageUnit *](#)) - input

The storage unit to be queried.

prop ([ODPropertyName](#)) - input

The property to be queried.

val ([ODValueType](#)) - input

The value to be queried.

name ([ODIText *](#)) - in/out

The IText structure of the specified storage unit, property and value. If this value is KODNULL, the existing buffer of text in IText is allocated and filled; otherwise, the buffer of text is disposed of and a new one is allocated and filled.

rv ([ODIText *](#)) - returns

The IText structure of the specified storage unit, property and value.

ODGetITextProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODGetMatrixProp

ODGetMatrixProp - Syntax

This function returns the matrix of the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODMatrix         *value;
ODMatrix         *rv;

rv = ODGetMatrixProp(ev, su, prop, val, value);
```

ODGetMatrixProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetMatrixProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetMatrixProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetMatrixProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetMatrixProp Parameter - value

value (ODMatrix *) - output
The matrix value from the specified storage unit, property and value.

ODGetMatrixProp Return Value - rv

rv (ODMatrix *) - returns
The matrix value from the specified storage unit, property and value.

ODGetMatrixProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

value (ODMatrix *) - output
The matrix value from the specified storage unit, property and value.

rv (ODMatrix *) - returns
The matrix value from the specified storage unit, property and value.

ODGetMatrixProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetPointProp

ODGetPointProp - Syntax

This function returns a point value from the specified storage unit, property and value.

```
#include <stdio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODPoint          *value;  
ODPoint          *rv;  
  
rv = ODGetPointProp(ev, su, prop, val, value);
```

ODGetPointProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetPointProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetPointProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetPointProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetPointProp Parameter - value

value ([ODPoint *](#)) - output
The point value from the specified storage unit, property and value.

ODGetPointProp Return Value - rv

rv ([ODPoint *](#)) - returns
The point value from the specified storage unit, property and value.

ODGetPointProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

value ([ODPoint *](#)) - output
The point value from the specified storage unit, property and value.

rv ([ODPoint *](#)) - returns
The point value from the specified storage unit, property and value.

ODGetPointProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetPolygonProp

ODGetPolygonProp - Syntax

This function returns a polygon value from the specified storage unit, property and value.

```
#include <stdio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODPolygon        *value;  
ODPolygon        *rv;
```

```
rv = ODGetPolygonProp(ev, su, prop, val, value);
```

ODGetPolygonProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetPolygonProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetPolygonProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetPolygonProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetPolygonProp Parameter - value

value ([ODPolygon *](#)) - output
The polygon value from the specified storage unit, property and value.

ODGetPolygonProp Return Value - rv

rv ([ODPolygon *](#)) - returns
The polygon value from the specified storage unit, property and value.

ODGetPolygonProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

value ([ODPolygon *](#)) - output
The polygon value from the specified storage unit, property and value.

rv ([ODPolygon *](#)) - returns
The polygon value from the specified storage unit, property and value.

ODGetPolygonProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetRectProp

ODGetRectProp - Syntax

This function returns a rectangle value from the specified storage unit, property and value.

```
#include <stdio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODRect           *value;  
ODRect           *rv;
```

```
rv = ODGetRectProp(ev, su, prop, val, value);
```

ODGetRectProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetRectProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetRectProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetRectProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetRectProp Parameter - value

value ([ODRect *](#)) - input
The rectangle value from the specified storage unit, property and value.

ODGetRectProp Return Value - rv

rv ([ODRect *](#)) - returns
The rectangle value from the specified storage unit, property and value.

ODGetRectProp - Parameters

ev ([Environment *](#)) - input

The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be queried.

prop (ODPropertyName) - input
The property to be queried.

val (ODValueType) - input
The value to be queried.

value (ODRect *) - input
The rectangle value from the specified storage unit, property and value.

rv (ODRect *) - returns
The rectangle value from the specified storage unit, property and value.

ODGetRectProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetSLongProp

ODGetSLongProp - Syntax

This function returns a signed long integer returned the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODSLong          rv;
```

```
rv = ODGetSLongProp(ev, su, prop, val);
```

ODGetSLongProp Parameter - ev

ev (Environment *) - input
The SOM Environment.

ODGetSLongProp Parameter - su

su (ODStorageUnit *) - input
The storage unit to be queried.

ODGetSLongProp Parameter - prop

prop (ODPropertyName) - input
The property to be queried.

ODGetSLongProp Parameter - val

val (ODValueType) - input
The value to be queried.

ODGetSLongProp Return Value - rv

rv (ODSLong) - returns
The signed-long integer from the specified storage unit, property and value.

ODGetSLongProp - Parameters

ev (Environment *) - input
The SOM Environment.

su (ODStorageUnit *) - input
The storage unit to be queried.

prop (ODPropertyName) - input
The property to be queried.

val (ODValueType) - input
The value to be queried.

rv (ODSLong) - returns
The signed-long integer from the specified storage unit, property and value.

ODGetSLongProp - Topics

Select an item:

ODGetSShortProp

ODGetSShortProp - Syntax

This function returns a singed short integer from the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODSShort         rv;
```

```
rv = ODGetSShortProp(ev, su, prop, val);
```

ODGetSShortProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetSShortProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetSShortProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetSShortProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetSShortProp Return Value - rv

rv ([ODSShort](#)) - returns
The signed-short integer read from the specified storage unit, property and value.

ODGetSShortProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

rv ([ODSShort](#)) - returns
The signed-short integer read from the specified storage unit, property and value.

ODGetSShortProp - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetStrongSUnitRefProp

ODGetStrongSUnitRefProp - Syntax

This function returns a strong storage unit reference from the specified storage unit, property and value.

```
#include <stdtypio.h>
```



```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODID             rv;  
  
rv = ODGetStrongSUNRefProp(ev, su, prop, val);
```

ODGetStrongSUNRefProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetStrongSUNRefProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetStrongSUNRefProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetStrongSUNRefProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetStrongSUNRefProp Return Value - rv

rv ([ODID](#)) - returns
The strong storage-unit reference from the storage unit, property and value. If the storage unit reference was not found, 0 is returned.

ODGetStrongSUNRefProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

rv ([ODID](#)) - returns
The strong storage-unit reference from the storage unit, property and value. If the storage unit reference was not found, 0 is returned.

ODGetStrongSURefProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODGetTime_TProp

ODGetTime_TProp - Syntax

This function returns a time data structure from the specified storage unit, property and value.

```
#include <stdttypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODTime           rv;
```

```
rv = ODGetTime_TProp(ev, su, prop, val);
```

ODGetTime_TProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetTime_TProp Parameter - su

su (ODStorageUnit *) - input
The storage unit to be queried.

ODGetTime_TProp Parameter - prop

prop (ODPropertyName) - input
The property to be queried.

ODGetTime_TProp Parameter - val

val (ODValueType) - input
The value to be queried.

ODGetTime_TProp Return Value - rv

rv (ODTime) - returns
The time data structure from the specified storage unit, property and value.

ODGetTime_TProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be queried.

prop (ODPropertyName) - input
The property to be queried.

val (ODValueType) - input
The value to be queried.

rv (ODTime) - returns
The time data structure from the specified storage unit, property and value.

ODGetTime_TProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODGetTypeListProp

ODGetTypeListProp - Syntax

This function returns a type list from the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODTypeList       *typeList;

ODGetTypeListProp(ev, su, prop, val, typeList);
```

ODGetTypeListProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetTypeListProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetTypeListProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetTypeListProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetTypeListProp Parameter - typeList

typeList ([ODTypeList *](#)) - output
The type list value from the specified storage unit, property and value. The property value containing 'n' elements has 'n+1' offsets. The first n offsets indicate the positions of the corresponding ISO string. The last offset equals the size of the value and is immediately before the first character of the first ISOstring. For example, a property value representing an empty type list is 4 bytes long and contains a single offset of value 4.

ODGetTypeListProp - Return Value

None.

ODGetTypeListProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

typeList ([ODTypeList *](#)) - output
The type list value from the specified storage unit, property and value. The property value containing 'n' elements has 'n+1' offsets. The first n offsets indicate the positions of the corresponding ISO string. The last offset equals the size of the value and is immediately before the first character of the first ISOstring. For example, a property value representing an empty type list is 4 bytes long and contains a single offset of value 4.

None.

ODGetTypeListProp - Topics

Select an item:

ODGetULongProp

ODGetULongProp - Syntax

This function returns an unsigned-long integer from the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODULong          rv;
```

```
rv = ODGetULongProp(ev, su, prop, val);
```

ODGetULongProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetULongProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetULongProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetULongProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetULongProp Return Value - rv

rv ([ODULong](#)) - returns
The unsigned-long integer from the specified storage unit, property and value.

ODGetULongProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

rv ([ODULong](#)) - returns
The unsigned-long integer from the specified storage unit, property and value.

ODGetULongProp - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetUShortProp

ODGetUShortProp - Syntax

This function returns a signed-short integer from the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODUShort         rv;  
  
rv = ODGetUShortProp(ev, su, prop, val);
```

ODGetUShortProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetUShortProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

ODGetUShortProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be queried.

ODGetUShortProp Parameter - val

val ([ODValueType](#)) - input
The value to be queried.

ODGetUShortProp Return Value - rv

rv ([ODUShort](#)) - returns
The signed-short integer from the specified storage unit, property and value.

ODGetUShortProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be queried.

prop ([ODPropertyName](#)) - input
The property to be queried.

val ([ODValueType](#)) - input
The value to be queried.

rv ([ODUShort](#)) - returns
The signed-short integer from the specified storage unit, property and value.

ODGetUShortProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODGetWeakSURefProp

ODGetWeakSURefProp - Syntax

This function returns a weak storage-unit reference from the specified storage unit, property and value.

```
#include <stdio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODID             rv;

rv = ODGetWeakSURefProp(ev, su, prop, val);
```

ODGetWeakSURefProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODGetWeakSURefProp Parameter - su

su (ODStorageUnit *) - input
The storage unit to be queried.

ODGetWeakSURefProp Parameter - prop

prop (ODPropertyName) - input
The property to be queried.

ODGetWeakSURefProp Parameter - val

val (ODValueType) - input
The value to be queried.

ODGetWeakSURefProp Return Value - rv

rv (ODID) - returns
The weak storage unit ID from the specified storage unit, property and value.

ODGetWeakSURefProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be queried.

prop (ODPropertyName) - input
The property to be queried.

val (ODValueType) - input
The value to be queried.

rv (ODID) - returns
The weak storage unit ID from the specified storage unit, property and value.

ODGetWeakSURefProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODSetBooleanProp

ODSetBooleanProp - Syntax

This function sets a boolean in the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODBoolean        value;
```

```
ODSetBooleanProp(ev, su, prop, val, value);
```

ODSetBooleanProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetBooleanProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetBooleanProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetBooleanProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetBooleanProp Parameter - value

value ([ODBoolean](#)) - input
The boolean value to set in the specified storage unit, property and value.

ODSetBooleanProp - Return Value

None.

ODSetBooleanProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODBoolean](#)) - input
The boolean value to set in the specified storage unit, property and value.

None.

ODSetBooleanProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetISOStrProp

ODSetISOStrProp - Syntax

This function sets an ISO string value in the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODISOStr         value;

ODSetISOStrProp(ev, su, prop, val, value);
```

ODSetISOStrProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetISOStrProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetISOStrProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetISOStrProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetISOStrProp Parameter - value

value ([ODISOStr](#)) - output
The ISO string to set in the specified storage unit, property and value.

ODSetISOStrProp - Return Value

None.

ODSetISOStrProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODISOStr](#)) - output
The ISO string to set in the specified storage unit, property and value.

None.

ODSetISOStrProp - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetITextProp

ODSetITextProp - Syntax

This function sets an IText structure in the specified storage unit, property and value.

```
#include <stdio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODIText          *name;

ODSetITextProp(ev, su, prop, val, name);
```

ODSetITextProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetITextProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetITextProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetITextProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetITextProp Parameter - name

name ([ODIText *](#)) - input
The IText structure to set in the specified storage unit, property and value.

ODSetITextProp - Return Value

None.

ODSetITextProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

name ([ODIText *](#)) - input
The IText structure to set in the specified storage unit, property and value.

None.

ODSetITextProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetMatrixProp

ODSetMatrixProp - Syntax

This function sets the matrix value in the specified storage unit, property and value.


```
#include <stdio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODMatrix         *value;

ODSetMatrixProp(ev, su, prop, val, value);
```

ODSetMatrixProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetMatrixProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetMatrixProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetMatrixProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetMatrixProp Parameter - value

value ([ODMatrix *](#)) - input
The matrix value to set in the specified storage unit, property and value.

ODSetMatrixProp - Return Value

None.

ODSetMatrixProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODMatrix *](#)) - input
The matrix value to set in the specified storage unit, property and value.

None.

ODSetMatrixProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetPointProp

ODSetPointProp - Syntax

This function sets the point value in the specified storage unit, property and value.

```
#include <stdio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODPoint          *value;
```

```
ODSetPointProp(ev, su, prop, val, value);
```

ODSetPointProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetPointProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetPointProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetPointProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetPointProp Parameter - value

value ([ODPoint *](#)) - output
The point value to set in the storage unit, property and value.

ODSetPointProp - Return Value

None.

ODSetPointProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODPoint *](#)) - output
The point value to set in the storage unit, property and value.

None.

ODSetPointProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetPolygonProp

ODSetPolygonProp - Syntax

This function sets the polygon value in the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
const ODPolygon  *value;

ODSetPolygonProp(ev, su, prop, val, value);
```

ODSetPolygonProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetPolygonProp Parameter - su

su (ODStorageUnit *) - input
The storage unit to be set.

ODSetPolygonProp Parameter - prop

prop (ODPropertyName) - input
The property to be set.

ODSetPolygonProp Parameter - val

val (ODValueType) - input
The value to be set.

ODSetPolygonProp Parameter - value

value (const ODPolygon *) - output
The polygon value to set in the storage unit, property and value.

ODSetPolygonProp - Return Value

None.

ODSetPolygonProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be set.

prop (ODPropertyName) - input

The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([const ODPolygon *](#)) - output
The polygon value to set in the storage unit, property and value.

None.

ODSetPolygonProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetRectProp

ODSetRectProp - Syntax

This function sets the rectangle value in the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODRect           *value;
```

```
ODSetRectProp(ev, su, prop, val, value);
```

ODSetRectProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetRectProp Parameter - su

su ([ODStorageUnit *](#)) - input

The storage unit to be set.

ODSetRectProp Parameter - prop

prop (ODPropertyName) - input
The property to be set.

ODSetRectProp Parameter - val

val (ODValueType) - input
The value to be set.

ODSetRectProp Parameter - value

value (ODRect *) - input
The rectangle value to set in the specified storage unit, property and value.

ODSetRectProp - Return Value

None.

ODSetRectProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be set.

prop (ODPropertyName) - input
The property to be set.

val (ODValueType) - input
The value to be set.

value (ODRect *) - input
The rectangle value to set in the specified storage unit, property and value.

None.

ODSetRectProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODSetSLongProp

ODSetSLongProp - Syntax

This function sets the signed-long value in the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODSLong          value;
```

```
ODSetSLongProp(ev, su, prop, val, value);
```

ODSetSLongProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetSLongProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetSLongProp Parameter - prop

prop (ODPropertyName) - input
The property to be set.

ODSetSLongProp Parameter - val

val (ODValueType) - input
The value to be set.

ODSetSLongProp Parameter - value

value (ODSLong) - input
The signed-long value to set in the specified storage unit, property and value.

ODSetSLongProp - Return Value

None.

ODSetSLongProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be set.

prop (ODPropertyName) - input
The property to be set.

val (ODValueType) - input
The value to be set.

value (ODSLong) - input
The signed-long value to set in the specified storage unit, property and value.

None.

ODSetSLongProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODSetSShortProp

ODSetSShortProp - Syntax

This function sets the signed-short value in the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODSShort         value;
```

```
ODSetSShortProp(ev, su, prop, val, value);
```

ODSetSShortProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetSShortProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetSShortProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetSShortProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetSShortProp Parameter - value

value ([ODSShort](#)) - input
The signed-short value to set in the specified storage unit, property and value.

ODSetSShortProp - Return Value

None.

ODSetSShortProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODSShort](#)) - input
The signed-short value to set in the specified storage unit, property and value.

None.

ODSetSShortProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetStrongSUNRefProp

ODSetStrongSUNRefProp - Syntax

This function sets a strong storage-unit reference in the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODID             id;

ODSetStrongSUNRefProp(ev, su, prop, val, id);
```

ODSetStrongSUNRefProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetStrongSUNRefProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetStrongSUNRefProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetStrongSUNRefProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetStrongSUNRefProp Parameter - id

id ([ODID](#)) - input
The strong storage unit ID to set in the specified storage unit, property and value.

ODSetStrongSUNRefProp - Return Value

None.

ODSetStrongSUNRefProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

id ([ODID](#)) - input
The strong storage unit ID to set in the specified storage unit, property and value.

None.

ODSetStrongSUNRefProp - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetTime_TProp

ODSetTime_TProp - Syntax

This function sets the time information in the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODTime           value;

ODSetTime_TProp(ev, su, prop, val, value);
```

ODSetTime_TProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetTime_TProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetTime_TProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetTime_TProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetTime_TProp Parameter - value

value ([ODTime](#)) - input
The time information to set in the specified storage unit, property and value.

ODSetTime_TProp - Return Value

None.

ODSetTime_TProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODTime](#)) - input
The time information to set in the specified storage unit, property and value.

None.

ODSetTime_TProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetTypeListProp

ODSetTypeListProp - Syntax

This function sets the type list value in the specified storage unit, property and value.

```
#include <stdio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODTypeList       *typeList;

ODSetTypeListProp(ev, su, prop, val, typeList);
```

ODSetTypeListProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetTypeListProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetTypeListProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetTypeListProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetTypeListProp Parameter - typeList

typeList ([ODTypeList *](#)) - input
The type list value to set in the specified storage unit, property and value.

ODSetTypeListProp - Return Value

None.

ODSetTypeListProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

typeList ([ODTypeList *](#)) - input
The type list value to set in the specified storage unit, property and value.

None.

ODSetTypeListProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetULongProp

ODSetULongProp - Syntax

This function sets the unsigned-long integer in the specified storage unit, property and value.

```
#include <stdint.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODULong          value;
```

```
ODSetULongProp(ev, su, prop, val, value);
```

ODSetULongProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetULongProp Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

ODSetULongProp Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be set.

ODSetULongProp Parameter - val

val ([ODValueType](#)) - input
The value to be set.

ODSetULongProp Parameter - value

value ([ODULong](#)) - input
The unsigned-long integer to set in the specified storage unit, property and value.

ODSetULongProp - Return Value

None.

ODSetULongProp - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODULong](#)) - input
The unsigned-long integer to set in the specified storage unit, property and value.

None.

ODSetULongProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetUShortProp

ODSetUShortProp - Syntax

This function sets the unsigned-short integer in the specified storage unit, property and value.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;
ODUShort         value;

ODSetUShortProp(ev, su, prop, val, value);
```

ODSetUShortProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetUShortProp Parameter - su

su (ODStorageUnit *) - input
The storage unit to be set.

ODSetUShortProp Parameter - prop

prop (ODPropertyName) - input
The property to be set.

ODSetUShortProp Parameter - val

val (ODValueType) - input
The value to be set.

ODSetUShortProp Parameter - value

value (ODUShort) - output
The unsigned short integer to set in the specified storage unit, property and value.

ODSetUShortProp - Return Value

None.

ODSetUShortProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be set.

prop (ODPropertyName) - input

The property to be set.

val ([ODValueType](#)) - input
The value to be set.

value ([ODUShort](#)) - output
The unsigned short integer to set in the specified storage unit, property and value.

None.

ODSetUShortProp - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSetWeakSUNRefProp

ODSetWeakSUNRefProp - Syntax

This function sets a weak storage-unit reference in the specified storage unit, property and value.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODID             id;
```

```
ODSetWeakSUNRefProp(ev, su, prop, val, id);
```

ODSetWeakSUNRefProp Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSetWeakSUNRefProp Parameter - su

su ([ODStorageUnit *](#)) - input

The storage unit to be set.

ODSetWeakSURefProp Parameter - prop

prop (ODPropertyName) - input
The property to be set.

ODSetWeakSURefProp Parameter - val

val (ODValueType) - input
The value to be set.

ODSetWeakSURefProp Parameter - id

id (ODID) - input
The weak storage-unit ID to set in the specified storage unit, property and value.

ODSetWeakSURefProp - Return Value

None.

ODSetWeakSURefProp - Parameters

ev (Environment *) - input
The SOM environment.

su (ODStorageUnit *) - input
The storage unit to be set.

prop (ODPropertyName) - input
The property to be set.

val (ODValueType) - input
The value to be set.

id (ODID) - input
The weak storage-unit ID to set in the specified storage unit, property and value.

None.

ODSetWeakSURefProp - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

ODSUAddPropValue

ODSUAddPropValue - Syntax

This function adds the property and value to the specified storage unit.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;
```

```
ODSUAddPropValue(ev, su, prop, val);
```

ODSUAddPropValue Parameter - ev

ev ([Environment](#) *) - input
The SOM environment.

ODSUAddPropValue Parameter - su

su (ODStorageUnit *) - in/out
The storage unit to be set.

ODSUAddPropValue Parameter - prop

prop ([ODPropertyName](#)) - input
The property to set in the storage unit..

ODSUAddPropValue Parameter - val

val ([ODValueType](#)) - input
The value to set in the storage unit..

ODSUAddPropValue - Return Value

None.

ODSUAddPropValue - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - in/out
The storage unit to be set.

prop ([ODPropertyName](#)) - input
The property to set in the storage unit..

val ([ODValueType](#)) - input
The value to set in the storage unit..

None.

ODSUAddPropValue - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSUForceFocus

ODSUForceFocus - Syntax

This function sets the focus to the specified property and value in a storage unit. If the property or value does not exist, it is created.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;
ODValueType      val;

ODSUForceFocus(ev, su, prop, val);
```

ODSUForceFocus Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSUForceFocus Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit whose focus is to be set.

ODSUForceFocus Parameter - prop

prop ([ODPropertyName](#)) - input
The property to focus to.

ODSUForceFocus Parameter - val

val ([ODValueType](#)) - input
The value to focus to.

ODSUForceFocus - Return Value

None.

ODSUForceFocus - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit whose focus is to be set.

prop ([ODPropertyName](#)) - input
The property to focus to.

val ([ODValueType](#)) - input
The value to focus to.

None.

ODSUForceFocus - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSUExistsThenFocus

ODSUExistsThenFocus - Syntax

This function sets the focus to the specified property and value only if they already exist in the specified storage unit.

```
#include <stdtypio.h>
```

```
Environment      *ev;  
ODStorageUnit    *su;  
ODPropertyName   prop;  
ODValueType      val;  
ODBoolean        rv;
```

```
rv = ODSUExistsThenFocus(ev, su, prop, val);
```

ODSUExistsThenFocus Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSUExistsThenFocus Parameter - su

su ([ODStorageUnit *](#)) - input
The storage unit whose focus is to be set.

ODSUExistsThenFocus Parameter - prop

prop ([ODPropertyName](#)) - input
The property to focus to.

ODSUExistsThenFocus Parameter - val

val ([ODValueType](#)) - input
The value to focus to.

ODSUExistsThenFocus Return Value - rv

rv ([ODBoolean](#)) - returns
A flag indicating whether the focus was successful.

kODTrue	The focus was successful.
kODFalse	The focus was unsuccessful.

ODSUExistsThenFocus - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit whose focus is to be set.

prop ([ODPropertyName](#)) - input

The property to focus to.

val ([ODValueType](#)) - input
The value to focus to.

rv ([ODBoolean](#)) - returns
A flag indicating whether the focus was successful.

kODTrue	The focus was successful.
kODFalse	The focus was unsuccessful.

ODSUExistsThenFocus - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

ODSURemoveProperty

ODSURemoveProperty - Syntax

This function removes the property of the specified storage unit.

```
#include <stdtypio.h>

Environment      *ev;
ODStorageUnit    *su;
ODPropertyName   prop;

ODSURemoveProperty(ev, su, prop);
```

ODSURemoveProperty Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

ODSURemoveProperty Parameter - su

su ([ODStorageUnit *](#)) - input

The storage unit whose property is to be removed.

ODSURemoveProperty Parameter - prop

prop ([ODPropertyName](#)) - input
The property to be removed.

ODSURemoveProperty - Return Value

None.

ODSURemoveProperty - Parameters

ev ([Environment *](#)) - input
The SOM environment.

su ([ODStorageUnit *](#)) - input
The storage unit whose property is to be removed.

prop ([ODPropertyName](#)) - input
The property to be removed.

None.

ODSURemoveProperty - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

Templates

The functions in this section are used to create and delete part templates in the OpenDoc Templates folder.

The following lists the template functions in alphabetic order.

- [ODCreateTemplate](#)
- [ODRemoveTemplate](#)

ODCreateTemplate

ODCreateTemplate - Syntax

This function creates a template from the specified part kind and editor and returns the template's Workplace Shell object ID.

```
#include <odtemps.h>

ODType      partKindName;
ODEditor     partEditor;
PSZ         rc;

rc = ODCreateTemplate(partKindName, partEditor);
```

ODCreateTemplate Parameter - partKindName

partKindName (ODType) - input
The part kind of the template.

ODCreateTemplate Parameter - partEditor

partEditor (ODEditor) - input
The part editor of the template.

ODCreateTemplate Return Value - rc

rc (PSZ) - returns
The Workplace Shell object ID for the new template.

ODCreateTemplate - Parameters

partKindName (ODType) - input
The part kind of the template.

partEditor ([ODEditor](#)) - input
The part editor of the template.

rc ([PSZ](#)) - returns
The Workplace Shell object ID for the new template.

ODCreateTemplate - Topics

Select an item:
[Syntax](#)
[Parameters](#)
[Returns](#)

ODRemoveTemplate

ODRemoveTemplate - Syntax

This function removes the template with the specified object ID.

```
#include <odtemps.h>
```

```
PSZ      ObjectID;  
LONG     rc;
```

```
rc = ODRemoveTemplate(ObjectID);
```

ODRemoveTemplate Parameter - ObjectID

ObjectID ([PSZ](#)) - input
The ID of the object whose template is to be removed.

ODRemoveTemplate Return Value - rc

rc ([LONG](#)) - returns
A flag indicating whether the template was successfully removed.

0	Successful completion.
1	Error occurred.

ODRemoveTemplate - Parameters

ObjectID ([PSZ](#)) - input

The ID of the object whose template is to be removed.

rc ([LONG](#)) - returns

A flag indicating whether the template was successfully removed.

0	Successful completion.
1	Error occurred.

ODRemoveTemplate - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

Window Properties

This section describes the functions used to get window properties from a frame.

The following lists the window properties functions in alphabetic order.

- [BeginGetWindowProperties](#)
- [EndGetWindowProperties](#)

BeginGetWindowProperties

BeginGetWindowProperties - Syntax

This function returns window properties from the storage unit referred to in the frame's kODPropWindowProperties.

```
#include <winutils.h>
```

```
Environment      *ev;  
ODFrame          *frame;  
WindowProperties *properties;  
ODBoolean        rc;
```

```
rc = BeginGetWindowProperties(ev, frame, properties);
```

BeginGetWindowProperties Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

BeginGetWindowProperties Parameter - frame

frame ([ODFrame *](#)) - input
The frame whose window properties are to be returned.

BeginGetWindowProperties Parameter - properties

properties ([WindowProperties *](#)) - output
The window properties of the specified frame.

BeginGetWindowProperties Return Value - rc

rc ([ODBoolean](#)) - returns
A flag indicating whether the annotation exist.

kODTrue	The annotation exists.
kODFalse	The annotation does not exist.

BeginGetWindowProperties - Parameters

ev ([Environment *](#)) - input
The SOM environment.

frame ([ODFrame *](#)) - input
The frame whose window properties are to be returned.

properties ([WindowProperties *](#)) - output
The window properties of the specified frame.

rc ([ODBoolean](#)) - returns
A flag indicating whether the annotation exist.

kODTrue	The annotation exists.
---------	------------------------

BeginGetWindowProperties - Topics

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

EndGetWindowProperties

EndGetWindowProperties - Syntax

This function releases the source frame.

```
#include <winutils.h>
```

```
Environment      *ev;  
WindowProperties *properties;
```

```
EndGetWindowProperties(ev, properties);
```

EndGetWindowProperties Parameter - ev

ev ([Environment *](#)) - input
The SOM environment.

EndGetWindowProperties Parameter - properties

properties ([WindowProperties *](#)) - input
The window properties of the source frame to be released.

EndGetWindowProperties - Return Value

None.

EndGetWindowProperties - Parameters

ev ([Environment *](#)) - input
The SOM environment.

properties ([WindowProperties *](#)) - input
The window properties of the source frame to be released.

None.

EndGetWindowProperties - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

Rexx Functions

This section describes the functions that can be called from REXX command files. Typically, these functions are used during the installation of OpenDoc part handlers or PM applications.

Note: Left and Right brackets ([]) are used to indicate optional parameters.

The following lists the REXX functions in alphabetic order.

Registration

- [RxODCreateTemplate](#)
- [RxODDeregisterPartHandler](#)
- [RxODDeregisterPartHandlerClass](#)
- [RxODLoadOpenDocFuncs](#)
- [RxODQueryPartHandlerInfo](#)
- [RxODQueryPartHandlerList](#)
- [RxODQueryPartKindInfo](#)
- [RxODQueryPartKindList](#)
- [RxODQueryPreferredPartHandler](#)
- [RxODQueryPreferredPartHandlerForCategory](#)
- [RxODQueryPreferredPartHandlerForFileExt](#)
- [RxODQueryPreferredPartHandlerForFileType](#)
- [RxODRegisterPartHandlerClass](#)
- [RxODRemoveTemplate](#)
- [RxODSetPreferredPartHandler](#)
- [RxODSetPreferredPartHandlerForCategory](#)
- [RxODSetPreferredPartHandlerForFileExt](#)
- [RxODSetPreferredPartHandlerForFileType](#)
- [RxODUnloadOpenDocFuncs](#)

Templates

- [RxODCreateTemplate](#)
- [RxODRemoveTemplate](#)

RxODCreateTemplate

RxODCreateTemplate - Purpose

This function creates a template from the specified part kind and editor and returns the template's Workplace Shell object ID.

RxODCreateTemplate Keyword - partKindName

partKindName
The part kind of the template.

RxODCreateTemplate Keyword - partEditor

partEditor
The part editor of the template.

RxODCreateTemplate - Keywords

partKindName
The part kind of the template.

partEditor
The part editor of the template.

RxODCreateTemplate - Syntax

RxODCreateTemplate ([partKindName](#) , [partEditor](#))

RxODCreateTemplate - Remarks

This function returns the Workplace Shell object ID for the new template.

This function is not loaded by the [RxODLoadOpenDocFuncs](#) function.

RxODCreateTemplate - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

[Remarks](#)

RxODDeregisterPartHandler

RxODDeregisterPartHandler - Purpose

This function deregisters a part handler with the OpenDoc registry and the OpenDoc SOM interface repository using the specified part handler name.

RxODDeregisterPartHandler Keyword - isoPartHandlerName

isoPartHandlerName

The name of the part handle to be deregistered.

RxODDeregisterPartHandler - Keywords

isoPartHandlerName

The name of the part handle to be deregistered.

RxODDeregisterPartHandler - Syntax

RxODDeregisterPartHandler ([isoPartHandlerName](#))

RxODDeregisterPartHandler - Remarks

This function returns one of the following return codes:

```
NO_ERROR  
ERROR_COULD_NOT_FIND_PART_HANDLER  
ERROR_COUNDTOT_FIND_CLASS  
ERROR_COUNDTOT_FIND_CLASS_IN_SOM_IR
```

RxODDeregisterPartHandler - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODDeregisterPartHandlerClass

RxODDeregisterPartHandlerClass - Purpose

This function deregisters a part handler with the OpenDoc registry and the OpenDoc SOM interface repository using the specified class name.

RxODDeregisterPartHandlerClass Keyword - pszClassName

pszClassName

The name of the class of the part handler to be deregistered.

RxODDeregisterPartHandlerClass - Keywords

pszClassName

The name of the class of the part handler to be deregistered.

RxODDeregisterPartHandlerClass - Syntax

RxODDeregisterPartHandlerClass ([pszClassName](#))

RxODDeregisterPartHandlerClass - Remarks

This function returns one of the following return codes:

NO_ERROR
ERROR_COULD_NOT_FIND_PART_HANDLER
ERROR_COUNND_NOT_FIND_CLASS
ERROR_COUNND_NOT_FIND_CLASS_IN_SOM_IR

RxODDeregisterPartHandlerClass - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODLoadOpenDocFuncs

RxODLoadOpenDocFuncs - Purpose

This function loads all part registration functions.

RxODLoadOpenDocFuncs - Keywords

There are no keywords for this command.

RxODLoadOpenDocFuncs - Syntax

RxODLoadOpenDocFuncs - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

RxODQueryPartHandlerInfo

RxODQueryPartHandlerInfo - Purpose

This function returns information about the specified part handler.

RxODQueryPartHandlerInfo Keyword - isoPartHandlerName

isoPartHandlerName

The part handler for which information is to be returned.

RxODQueryPartHandlerInfo Keyword - [partHandlerDisplayName]

[partHandlerDisplayName]

The display name of the part handler.

RxODQueryPartHandlerInfo Keyword - [partHandlerClassName]

[partHandlerClassName]

The class name of the part handler.

RxODQueryPartHandlerInfo Keyword - [partKindName]

[partKindName]

The part kinds supported by the part handler.

RxODQueryPartHandlerInfo Keyword - [ole2ClassId]

[ole2ClassId]

The OLE class ID for the part handler.

RxODQueryPartHandlerInfo Keyword - [windowIconFileName]

[windowIconFileName]

The file name of the Windows icon.

RxODQueryPartHandlerInfo Keyword - [DLLName]

[DLLName]

The DLL file name of the part handler.

RxODQueryPartHandlerInfo - Keywords

isoPartHandlerName

The part handler for which information is to be returned.

[partHandlerDisplayName]

The display name of the part handler.

[partHandlerClassName]

The class name of the part handler.

[partKindName]

The part kinds supported by the part handler.

[ole2ClassId]

The OLE class ID for the part handler.

[windowIconFileName]

The file name of the Windows icon.

[DLLName]

The DLL file name of the part handler.

RxODQueryPartHandlerInfo - Syntax

```
RxODQueryPartHandlerInfo ( [isoPartHandlerName] , [partHandlerDisplayName] ,  
    [partHandlerClassName] , [partKindName] ,  
    [ole2ClassId] , [windowIconFileName] ,  
    [DLLName] )
```

RxODQueryPartHandlerInfo - Remarks

This function returns one of the following return codes:

```
NO_ERROR  
errPR_BUFFER_TOO_SMALL  
ERROR_COULD_NOT_FIND_PART_HANDLER
```

RxODQueryPartHandlerInfo - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODQueryPartHandlerList

RxODQueryPartHandlerList - Purpose

This function returns a list of part handler names that meet the given criteria. If the part kind and category are not specified, a list of all part handlers is returned.

RxODQueryPartHandlerList Keyword - Editors

Editors

The REXX stem variable to receive the list of part handlers.

RxODQueryPartHandlerList Keyword - [isoPartKind]

[isoPartKind]

The part kind that the part handlers can edit. This parameter is optional and can be set to kODNULL.

RxODQueryPartHandlerList Keyword - [pszCategory]

[pszCategory]

The category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to kODNULL.

RxODQueryPartHandlerList - Keywords

Editors

The REXX stem variable to receive the list of part handlers.

[isoPartKind]

The part kind that the part handlers can edit. This parameter is optional and can be set to kODNULL.

[pszCategory]

The category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to kODNULL.

RxODQueryPartHandlerList - Syntax

RxODQueryPartHandlerList ([Editors](#) , [isoPartKind], [\[pszCategory\]](#))

RxODQueryPartHandlerList - Remarks

This function returns one of the following return codes:

NO_ERROR
errPR_BUFFER_TOO_SMALL

RxODQueryPartHandlerList - Topics

Select an item:

[Purpose](#)

RxODQueryPartKindInfo

RxODQueryPartKindInfo - Purpose

This function returns information about the specified part kind.

RxODQueryPartKindInfo Keyword - isoPartHandlerName

isoPartHandlerName

The part handler name for which information is to be returned.

RxODQueryPartKindInfo Keyword - isoPartKindName

isoPartKindName

The part kind for which information is to be returned.

RxODQueryPartKindInfo Keyword - partKindQueryInfo

partKindQueryInfo

A structure containing information about the part kind.

RxODQueryPartKindInfo - Keywords

isoPartHandlerName

The part handler name for which information is to be returned.

isoPartKindName

The part kind for which information is to be returned.

partKindQueryInfo

A structure containing information about the part kind.

RxODQueryPartKindInfo - Syntax

RxODQueryPartKindInfo ([isoPartHandlerName](#) , [isoPartKindName](#) , [partKindQueryInfo](#))

RxODQueryPartKindInfo - Remarks

This function returns one of the following return codes:

```
NO_ERROR
errPR_BUFFER_TOO_SMALL
ERROR_COULD_NOT_FIND_PART_HANDLER
```

RxODQueryPartKindInfo - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODQueryPartKindList

RxODQueryPartKindList - Purpose

This function returns a list of part kinds that meets the given criteria. If the part handler and category are not specified, a list of all part-registered part kinds is returned.

RxODQueryPartKindList Keyword - Kinds

Kinds

The REXX stem variable to receive a list of part kinds.

RxODQueryPartKindList Keyword - [isoPartHandlerName]

[isoPartHandlerName]

The part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

RxODQueryPartKindList Keyword - [pszCategory]

[pszCategory]

The category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

RxODQueryPartKindList - Keywords

Kinds

The REXX stem variable to receive a list of part kinds.

[isoPartHandlerName]

The part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

[pszCategory]

The category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

RxODQueryPartKindList - Syntax

RxODQueryPartKindList (Kinds, [\[isoPartHandlerName\]](#) , [\[pszCategory\]](#))

RxODQueryPartKindList - Remarks

This function returns one of the following return codes:

NO_ERROR
errPR_BUFFER_TOO_SMALL

RxODQueryPartKindList - Topics

Select an item:

RxODQueryPreferredPartHandler

RxODQueryPreferredPartHandler - Purpose

This function returns the preferred part-handler name for the specified part kind.

RxODQueryPreferredPartHandler Keyword - isoPartKindName

isoPartKindName
The part kind name.

RxODQueryPreferredPartHandler Keyword - isoPartHandlerName

isoPartHandlerName
The returned preferred part handler for the specified part kind.

RxODQueryPreferredPartHandler - Keywords

isoPartKindName
The part kind name.

isoPartHandlerName
The returned preferred part handler for the specified part kind.

RxODQueryPreferredPartHandler - Syntax

RxODQueryPreferredPartHandler ([isoPartKindName](#) , [isoPartHandlerName](#))

RxODQueryPreferredPartHandler - Remarks

This function returns one of the following return codes:

NO_ERROR
errPR_BUFFER_TOO_SMALL
ERROR_COULD_NOT_FIND_PART_KIND

RxODQueryPreferredPartHandler - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODQueryPreferredPartHandlerForCategory

RxODQueryPreferredPartHandlerForCategory - Purpose

This function returns information about the preferred part-handler using the specified category.

RxODQueryPreferredPartHandlerForCategory Keyword - isoCategory

isoCategory
The category of the part.

RxODQueryPreferredPartHandlerForCategory Keyword - isoPartHa

isoPartHandlerName
The returned preferred part handler for the specified category.

RxODQueryPreferredPartHandlerForCategory Keyword - pulBufferS

pulBufferSize

On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the list of part kinds placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

RxODQueryPreferredPartHandlerForCategory - Keywords

isoCategory

The category of the part.

isoPartHandlerName

The returned preferred part handler for the specified category.

pulBufferSize

On input, this parameter is the size, in bytes, of the specified buffer. On output, the actual size of the list of part kinds placed in the buffer is returned. If the buffer is too small, the size of the buffer required to hold the string is returned.

RxODQueryPreferredPartHandlerForCategory - Syntax

```
RxODQueryPreferredPartHandlerForCategory ( isoCategory , isoPartHandlerName ,  
                                           pulBufferSize )
```

RxODQueryPreferredPartHandlerForCategory - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

RxODQueryPreferredPartHandlerForFileExt

RxODQueryPreferredPartHandlerForFileExt - Purpose

This function returns information about the preferred part-handler using the specified file extension.

RxODQueryPreferredPartHandlerForFileExt Keyword - isoFileExt

isoFileExt

The file extension of the part.

RxODQueryPreferredPartHandlerForFileExt Keyword - isoPartHand

isoPartHandlerName

The returned preferred part handler for the specified file extension.

RxODQueryPreferredPartHandlerForFileExt - Keywords

isoFileExt

The file extension of the part.

isoPartHandlerName

The returned preferred part handler for the specified file extension.

RxODQueryPreferredPartHandlerForFileExt - Syntax

RxODQueryPreferredPartHandlerForFileExt ([isoFileExt](#) , [isoPartHandlerName](#))

RxODQueryPreferredPartHandlerForFileExt - Remarks

This function returns one of the following return codes:

NO_ERROR
errPR_BUFFER_TOO_SMALL

RxODQueryPreferredPartHandlerForFileExt - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODQueryPreferredPartHandlerForFileType

RxODQueryPreferredPartHandlerForFileType - Purpose

This function returns the preferred part-handler name for the specified file type.

RxODQueryPreferredPartHandlerForFileType Keyword - partFileType

partFileType
The file type of the part.

RxODQueryPreferredPartHandlerForFileType Keyword - isoPartHandlerName

isoPartHandlerName
The returned preferred part handler for the specified file type.

RxODQueryPreferredPartHandlerForFileType - Keywords

partFileType
The file type of the part.

isoPartHandlerName
The returned preferred part handler for the specified file type.

RxODQueryPreferredPartHandlerForFileType - Syntax

RxODQueryPreferredPartHandlerForFileType ([partFileType](#) , [isoPartHandlerName](#))

RxODQueryPreferredPartHandlerForFileType - Remarks

This function returns one of the following return codes:

NO_ERROR'.
errPR_BUFFER_TOO_SMALL'.

RxODQueryPreferredPartHandlerForFileType - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODRegisterPartHandlerClass

RxODRegisterPartHandlerClass - Purpose

This function registers a part handler with the OpenDoc registry.

RxODRegisterPartHandlerClass Keyword - pszClassName

pszClassName

The name of the SOM class to be registered.

RxODRegisterPartHandlerClass Keyword - [pszDllName]

[pszDllName]

The qualified or unqualified DLL name of the class. The default DLL name is the same as the class name.

RxODRegisterPartHandlerClass Keyword - [cTemplate]

[cTemplate]

A flag indicating whether a template should be created.

RxODRegisterPartHandlerClass Keyword - [reserved]

[reserved]
Reserved value.

RxODRegisterPartHandlerClass - Keywords

pszClassName
The name of the SOM class to be registered.

[pszDllName]
The qualified or unqualified DLL name of the class. The default DLL name is the same as the class name.

[cTemplate]
A flag indicating whether a template should be created.

[reserved]
Reserved value.

RxODRegisterPartHandlerClass - Syntax

```
RxODRegisterPartHandlerClass ( pszClassName , \[pszDllName\] , \[cTemplate\],  
                                \[reserved\] ).
```

RxODRegisterPartHandlerClass - Remarks

This function returns one of the following return codes:

```
NO_ERROR  
errPR_INVALID_CLASSNAME  
errPR_COULD_NOT_LOAD_CLASS  
errPR_NOT_PART_HANDLER_CLASS  
errPR_PART_HANDLER_INFO_REPLACED  
errPR_PART_HANDLER_NOT_FOUND  
errPR_PART_KIND_NOT_FOUND  
errPR_BUFFER_TOO_SMALL
```

This function creates an instance of the specified class using the specified DLL. The successful completion of this function depends on the information that is provided by the part-handler class. If the handler has already been registered, this function replaces the handler information.

RxODRegisterPartHandlerClass - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

[Remarks](#)

RxODRemoveTemplate

RxODRemoveTemplate - Purpose

This function removes the template with the specified object ID.

RxODRemoveTemplate Keyword - ObjectID

ObjectID

The ID of the object whose template is to be removed.

RxODRemoveTemplate - Keywords

ObjectID

The ID of the object whose template is to be removed.

RxODRemoveTemplate - Syntax

RxODRemoveTemplate ([ObjectID](#))

RxODRemoveTemplate - Remarks

This function returns one of the following flags, indicating whether the template was successfully removed.

0	Successful completion.
1	Error occurred.

This function is not loaded by the [RxODLoadOpenDocFuncs](#) function.

RxODRemoveTemplate - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

[Remarks](#)

RxODSetPreferredPartHandler

RxODSetPreferredPartHandler - Purpose

This function sets the preferred part-handler name.

RxODSetPreferredPartHandler Keyword - isoPartKindName

isoPartKindName

The part kind name.

RxODSetPreferredPartHandler Keyword - isoPartHandleName

isoPartHandleName

The part handler name.

RxODSetPreferredPartHandler - Keywords

isoPartKindName

The part kind name.

isoPartHandleName

The part handler name.

RxODSetPreferredPartHandler - Syntax

RxODSetPreferredPartHandler ([isoPartKindName](#) , isoPartHandleName)

RxODSetPreferredPartHandler - Remarks

This function returns one of the following return codes:

NO_ERROR
ERROR_COULD_NOT_FIND_PART_KIND

RxODSetPreferredPartHandler - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODSetPreferredPartHandlerForCategory

RxODSetPreferredPartHandlerForCategory - Purpose

This function sets the preferred part-handler to the specified name using the specified category.

RxODSetPreferredPartHandlerForCategory Keyword - isoCategory

isoCategory
The category of the part.

RxODSetPreferredPartHandlerForCategory Keyword - isoPartHandl

isoPartHandlerName
The part handler name.

RxODSetPreferredPartHandlerForCategory - Keywords

isoCategory
The category of the part.

isoPartHandlerName
The part handler name.

RxODSetPreferredPartHandlerForCategory - Syntax

RxODSetPreferredPartHandlerForFileCategory ([isoCategory](#) , [isoPartHandlerName](#))

RxODSetPreferredPartHandlerForCategory - Remarks

This function returns one of the following return codes:

NO_ERROR
ERROR_COULD_NOT_FIND_PART_KIND

RxODSetPreferredPartHandlerForCategory - Topics

Select an item:

[Purpose](#)
[Syntax](#)
[Keywords](#)
[Remarks](#)

RxODSetPreferredPartHandlerForFileExt

RxODSetPreferredPartHandlerForFileExt - Purpose

This function sets the preferred part-handler to the specified name using the specified file extension.

RxODSetPreferredPartHandlerForFileExt Keyword - isoFileExt

isoFileExt

The file extension of the part.

RxODSetPreferredPartHandlerForFileExt Keyword - isoPartHandler

isoPartHandlerName

The part handler name.

RxODSetPreferredPartHandlerForFileExt - Keywords

isoFileExt

The file extension of the part.

isoPartHandlerName

The part handler name.

RxODSetPreferredPartHandlerForFileExt - Syntax

RxODSetPreferredPartHandlerForFileExt ([isoFileExt](#) , [isoPartHandlerName](#))

RxODSetPreferredPartHandlerForFileExt - Remarks

This function returns one of the following return codes:

NO_ERROR

RxODSetPreferredPartHandlerForFileExt - Topics

Select an item:

RxODSetPreferredPartHandlerForFileType

RxODSetPreferredPartHandlerForFileType - Purpose

This function sets the preferred part-handler to the specified name using the specified file type.

RxODSetPreferredPartHandlerForFileType Keyword - isoFileType

isoFileType
The file type name.

RxODSetPreferredPartHandlerForFileType Keyword - isoPartHandlerName

isoPartHandlerName
The part handler name.

RxODSetPreferredPartHandlerForFileType - Keywords

isoFileType
The file type name.

isoPartHandlerName
The part handler name.

RxODSetPreferredPartHandlerForFileType - Syntax

RxODSetPreferredPartHandlerForFileType ([isoFileType](#) , [isoPartHandlerName](#))

RxODSetPreferredPartHandlerForFileType - Remarks

This function returns one of the following return code:

NO_ERROR

RxODSetPreferredPartHandlerForFileType - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

[Remarks](#)

RxODUnloadOpenDocFuncs

RxODUnloadOpenDocFuncs - Purpose

This function unloads all part registration functions.

RxODUnloadOpenDocFuncs - Keywords

There are no keywords for this command.

RxODUnloadOpenDocFuncs - Syntax

RxODUnloadd ()

RxODUnloadOpenDocFuncs - Topics

Select an item:

[Purpose](#)

[Syntax](#)

[Keywords](#)

Presentation Manager Messages Related to OpenDoc

This section contains an alphabetic list of the messages that can be processed by windows and dialog procedures within OpenDoc documents.

OD_HELP

OD_HELP - Syntax

This message occurs when the user requests contextual help on either a menu item or a control field within the part.

```
param1
    ODUShort  usContext

param2
    ODUShort  usTopic
    ODUShort  usSubTopic

returns
    BOOL      rc
```

OD_HELP Field - usContext

usContext ([ODUShort](#))

The type of window on which help was requested. This field can be set to one of the following values:

HLPM_WINDOW	Application window
HLPM_FRAME	Frame window
HLPM_MENU	Part menu window
HLPM_ODMENU	Document shell menu window

OD_HELP Field - usTopic

usTopic (ODUShort)

The topic identifier. If *usContext* has a value of HLPM_WINDOW or HLPM_FRAME, this field contains the ID of the window containing the control field on which help was requested. If *usContext* has a value of HLPM_MENU, this field contains the ID of the submenu containing the control field on which help was requested. If *usContext* has a value of HLPM_ODMENU or HLPM_MENU, this field contains the ID of the menu on which help was requested.

OD_HELP Field - usSubTopic

usSubTopic (ODUShort)

The subtopic identifier. If *usContext* has a value of HLPM_WINDOW or HLPM_FRAME, this field contains the control ID of the cursored field on which help was requested. If *usContext* has a value of HLPM_ODMENU or HLPM_MENU, this field contains the ID of the submenu item on which help was requested. If help was requested on a top level menu, this field will contain a-1.

OD_HELP Return Value - rc

rc (BOOL)

Processed indicator.

TRUE

Message was processed.

FALSE

Message was ignored.

OD_HELP - Parameters

usContext (ODUShort)

The type of window on which help was requested. This field can be set to one of the following values:

HLPM_WINDOW

Application window

HLPM_FRAME

Frame window

HLPM_MENU

Part menu window

HLPM_ODMENU

Document shell menu window

usTopic (ODUShort)

The topic identifier. If *usContext* has a value of HLPM_WINDOW or HLPM_FRAME, this field contains the ID of the window containing the control field on which help was requested. If *usContext* has a value of HLPM_MENU, this field contains the ID of the submenu containing the control field on which help was requested. If *usContext* has a value of HLPM_ODMENU or HLPM_MENU, this field contains the ID of the menu on which help was requested.

usSubTopic (ODUShort)

The subtopic identifier. If *usContext* has a value of HLPM_WINDOW or HLPM_FRAME, this field contains the control ID of the cursored field on which help was requested. If *usContext* has a value of HLPM_ODMENU or HLPM_MENU, this field contains the ID of the submenu item on which help was requested. If help was requested on a top level menu, this field will contain a-1.

rc (BOOL)

Processed indicator.

TRUE	Message was processed.
FALSE	Message was ignored.

OD_HELP - Remarks

Parts should not create help subtables for their contextual help, but instead process the OD_HELP message. If a part consists of an embedded control or dialog with its own menu and the user presses **F1** on a field, *param1* contains HLPM_WINDOW or HLPM_FRAME, depending on how the control or dialog was created. If the user presses **F1** on the part's dialog menu, then *param1* contains HLPM_MENU. If the user presses **F1** on a part-defined menu item of the document shell window, *param1* contains HLPM_ODMENU.

OD_HELP - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)
[Remarks](#)

OD_PRINT

OD_PRINT - Syntax

This OpenDoc message occurs when the **Print** menu item is selected from the **Document** menu of the window, when the shell is invoked with the -print command line option, or when the shell processes the OSA "Print Document" event.

```

param1
    PSZ    destPrintQueue

param2
    ULONG  ulReserved

returns
    ULONG  ulReserved

```

OD_PRINT Field - destPrintQueue

destPrintQueue (PSZ)

If the document shell was invoked with the -print=QUEUENAME command line option, this parameter points to the specified queue

name. The part handler should validate the name and display the specified print queue as the selected queue in the print dialog.

If the queue name is not specified, the system default queue is used.

OD_PRINT Field - ulReserved

ulReserved ([ULONG](#))
Reserved value.

OD_PRINT Return Value - ulReserved

ulReserved ([ULONG](#))
Reserved value. This parameter should be set to 0 by the part handler.

OD_PRINT - Parameters

destPrintQueue ([PSZ](#))
If the document shell was invoked with the -print=QUEUENAME command line option, this parameter points to the specified queue name. The part handler should validate the name and display the specified print queue as the selected queue in the print dialog.

If the queue name is not specified, the system default queue is used.

ulReserved ([ULONG](#))
Reserved value.

ulReserved ([ULONG](#))
Reserved value. This parameter should be set to 0 by the part handler.

OD_PRINT - Topics

Select an item:

[Syntax](#)
[Parameters](#)
[Returns](#)

Data Types

The following scalar types, enumerations and structures are used in the OpenDoc interface. They are listed in alphabetic order.

The following naming conventions can help you understand the function of the various OpenDoc identifiers.

- Names of OpenDoc data types, including structures and enumerations, begin with the prefix OD.

- Names of OpenDoc constants begin with the prefix kOD.
- The names of certain constants include a type prefix after the standard kOD prefix. For example, constant position codes begin with kODPos, where kOD is the standard constant prefix and Pos indicates that the constant specifies a position code.
- Names of other constants include a type suffix. For example, the constant transform types end with the suffix Xform.

NULL Constants Used in OpenDoc

The following constant is used throughout OpenDoc:

kODNULL A 32-bit value representing null. This constant can be used for a null reference to an OpenDoc object of any class and for null values of most 32-bit data types.

The following constants must be used instead for null values of the indicated types:

kODNullTypeToken	For the ODTypeToken type
kODNullFocus	For a null tokenized focus type
kODNULLKey	For the ODStorageUnitKey type
kODNULLID	For the types ODID , ODDocumentID , ODDraftID , and ODStorageUnitID
kODNoEditor	For the ODEditor type

_IDL_SEQUENCE_PartKindInfo

A structure representing the part kind data for a part handler.

```
typedef struct __IDL_SEQUENCE_PartKindInfo {
    unsigned long    _maximum;
    unsigned long    _length;
    PartKindInfo     *_buffer;
} _IDL_SEQUENCE_PartKindInfo;
```

_IDL_SEQUENCE_PartKindInfo Field - **_maximum**

_maximum (unsigned long)
The maximum size of *_buffer* .

_IDL_SEQUENCE_PartKindInfo Field - **_length**

_length (unsigned long)
The actual length of the data currently being passed in *_buffer* .

`_IDL_SEQUENCE_PartKindInfo` Field - `_buffer`

`_buffer` (`PartKindInfo *`)
An array of part kind info structures.

`_IDL_SEQUENCE_octet`

A structure representing foreign data larger than 4 bytes, or variable-length data in general.

```
typedef struct __IDL_SEQUENCE_octet {  
    unsigned long    _maximum;  
    unsigned long    _length;  
    octet            *_buffer;  
} _IDL_SEQUENCE_octet;
```

`_IDL_SEQUENCE_octet` Field - `_maximum`

`_maximum` (unsigned long)
The size (in bytes) of the memory block containing data and thus the maximum size of data that can be stored there.

`_IDL_SEQUENCE_octet` Field - `_length`

`_length` (unsigned long)
The length (number of bytes) of the data currently in the memory block.

`_IDL_SEQUENCE_octet` Field - `_buffer`

`_buffer` (octet *)
A pointer to the memory block containing the data.

`_IDL_SEQUENCE_string`

A structure representing data which can contain all possible 8-bit quantities.

```
typedef struct __IDL_SEQUENCE_string {
```

```
unsigned long    _maximum;  
unsigned long    _length;  
string          *_buffer;  
} _IDL_SEQUENCE_string;
```

_IDL_SEQUENCE_string Field - _maximum

_maximum (unsigned long)
The maximum size of *_buffer* .

_IDL_SEQUENCE_string Field - _length

_length (unsigned long)
The actual length of the data currently being passed in *_buffer* .

_IDL_SEQUENCE_string Field - _buffer

_buffer (string *)
An array of strings.

ACCEL

Accelerator structure.

```
typedef struct _ACCEL {  
    USHORT    fs; /* Options. */  
    USHORT    key; /* Key. */  
    USHORT    cmd; /* Command code. */  
} ACCEL;
```

```
typedef ACCEL *PACCEL;
```

ACCEL Field - fs

fs (USHORT)
Options.

ACCEL Field - key

key ([USHORT](#))
Key.

ACCEL Field - cmd

cmd ([USHORT](#))
Command code.

The value to be placed in the *uscmd* parameter of a WM_HELP, a WM_COMMAND, or a WM_SYSCOMMAND.

AEDesc

Descriptor record structure.

```
typedef struct _AEDesc {  
    DescType    descriptorType;  
    Handle      dataHandle;  
} AEDesc;
```

In OS/2, this structure must not be accessed directly by the programmer. The AEClearDesc, AEGetDescData, and AESizeOfDescData functions can be used to access the fields of an OS/2 AEDesc structure. The AECreatDesc function is used to create a new descriptor and initialize it with a type and data.

AEDesc Field - descriptorType

descriptorType ([DescType](#))
A four-character string of type [DescType](#) that indicates the type of data being passed.

AEDesc Field - dataHandle

dataHandle ([Handle](#))
A handle to the data being passed.

AEDescList

List of descriptor records.

```
typedef AEDesc AEDescList;
```

AEEventClass

A 4-character code representing the event class of an OSA event.

```
typedef ODSLong AEEventClass;
```

AEEventID

An OSA event ID, for example, kAEGetData, the Get Data event.

```
typedef unsigned long AEEventID;
```

AEKeyword

A constant value that indicates a particular type of descriptor record. See *OSA Event Registry: Standard Suites* for more information about keywords.

```
typedef unsigned long AEKeyword;
```

AERecord

List of keyword-specific descriptor records.

```
typedef AEDescList AERecord;
```

AESystemHandler

System handler.

```
typedef struct _AESystemHandler {  
    char      *moduleName;  
    char      *procedureName;  
} AESystemHandler;  
  
typedef AESystemHandler *pAESystemHandler;
```

The specified module and procedure names must be usable in the DosLoadModule and DosGetProcAddr functions.

AESystemHandler Field - moduleName

moduleName (char *)
Pointer to the DLL name.

AESystemHandler Field - procedureName

procedureName (char *)
Pointer to the procedure entry-point name.

APIRET

Unsigned integer in the range 0 through 4 294 967 295.

```
typedef unsigned long APIRET;
```

BOOL

Boolean.

Valid values are:

- FALSE, which is 0
- TRUE, which is 1

```
typedef unsigned long BOOL;
```

Boolean

Boolean.

```
typedef boolean Boolean;
```

Valid values are FALSE, which is 0, and TRUE, which is 1.

CallbackCallerProc

A pointer to a function.

```
typedef void *CallbackCallerProc;
```

CDATE

Structure that contains date information for a data element in the details view of a container control.

```
typedef struct _CDATE {  
    UCHAR    day;      /* Current day. */  
    UCHAR    month;    /* Current month. */  
    USHORT   year;     /* Current year. */  
} CDATE;
```

```
typedef CDATE *PCDATE;
```

CDATE Field - day

day (UCHAR)
Current day.

CDATE Field - month

month (UCHAR)
Current month.

CDATE Field - year

year (USHORT)
Current year.

CTIME

Structure that contains time information for a data element in the details view of a container control.

```
typedef struct _CTIME {  
    UCHAR    hours;    /* Current hour. */  
    UCHAR    minutes;  /* Current minute. */  
    UCHAR    seconds;  /* Current second. */  
}
```

```
    UCHAR    ucReserved; /* Reserved. */  
} CTIME;  
  
typedef CTIME *PCTIME;
```

CTIME Field - hours

hours (UCHAR)
Current hour.

CTIME Field - minutes

minutes (UCHAR)
Current minute.

CTIME Field - seconds

seconds (UCHAR)
Current second.

CTIME Field - ucReserved

ucReserved (UCHAR)
Reserved.

DescType

Descriptor type.

```
typedef unsigned long DescType;
```

Descriptor types represent various data types. See *OSA Event Registry: Standard Suites* for more information about descriptor types.

DEVOPENSTRUC

Open-device data structure.

```
typedef struct _DEVOPENSTRUC {
    PSZ      pszLogAddress;      /* Logical address. */
    PSZ      pszDriverName;      /* Driver name. */
    PDRIVDATA pdriv;             /* Driver data. */
    PSZ      pszDataType;        /* Data type. */
    PSZ      pszComment;         /* Comment. */
    PSZ      pszQueueProcName;    /* Queue-processor name. */
    PSZ      pszQueueProcParams; /* Queue-processor parameters. */
    PSZ      pszSpoolerParams;    /* Spooler parameters. */
    PSZ      pszNetworkParams;    /* Network parameters. */
} DEVOPENSTRUC;

typedef DEVOPENSTRUC *PDEVOPENSTRUC;
```

DEVOPENSTRUC Field - pszLogAddress

pszLogAddress (PSZ)

Logical address.

This is required for an OD_DIRECT device being opened with DevOpenDC; it is the logical device address, such as "LPT1" on OS/2. Some drivers may accept a file name for this parameter or even a named pipe.

Where output is to be queued (for an OD_QUEUED device), this is the name of the queue for the output device. The queue name can be a UNC name.

Note: This parameter can be a port name for a printer device context.

DEVOPENSTRUC Field - pszDriverName

pszDriverName (PSZ)

Driver name.

Character string identifying the printer driver, for example, LASERJET. The Default Device Driver field of the PRQINFO3 structure, associated with the required print queue, gives the driver and device name, separated by a period, for example LASERJET . HP LaserJet IIID . It can contain only the name up to the period, for example LASERJET.

DEVOPENSTRUC Field - pdriv

pdriv (PDRIVDATA)

Driver data.

Data that is to be passed directly to the PM device driver. Whether any of this is required depends upon the device driver.

For printer device context, this is a pointer to the job properties data.

DEVOPENSTRUC Field - pszDataType

pszDataType (PSZ)

Data type.

For an OD_QUEUED or OD_DIRECT device, this parameter defines the type of data that is to be queued as follows:

PM_Q_STD	Standard format
PM_Q_RAW	Raw format

Note that a device driver can define other data types.

For OD_QUEUED or OD_DIRECT device types, the default is supplied by the device driver if *pszDataType* is not specified. For any other device type, *pszDataType* is ignored.

DEVOPENSTRUC Field - pszComment

pszComment (PSZ)

Comment.

Optional character string that the printer object displays to the user in a job settings notebook. It is recommended that the application include its own name in this comment string.

Note: The job title text is derived from the document name passed to DevEscape (DEVESC_STARTDOC).

DEVOPENSTRUC Field - pszQueueProcName

pszQueueProcName (PSZ)

Queue-processor name.

This is the name of the queue processor, for queued output, and is usually the default.

DEVOPENSTRUC Field - pszQueueProcParams

pszQueueProcParams (PSZ)

Queue-processor parameters.

Queue processor parameters (optional). They can include information such as the number of copies you want to print and the size of the output area on the printed page.

The first parameter (*COP*) is used for all spool-file formats. The remaining parameters are valid for PM_Q_STD spool files only. Because PM_Q_STD data are used mainly for *graphic* data, these parameters are described in relation to the printing of picture files.

The PMPRINT/PMPLLOT queue-processor parameters are separated by spaces and are:

COP=*n*

The *COP* parameter specifies the number of copies of the spool file that you want printed. The value of *n* must be an integer in the range of 1 through 999.

The default is *COP=1*.

ARE=C | *w,h,l,t*

The *ARE* parameter determines the size and position of the output area. This is the area of the physical page to

which printing is restricted.

The default value of *ARE=C* means that the output area is the whole page. Note, however, that the printer cannot print outside its own device clip limits.

To size and position the output area at a specific point on the page, use *ARE=w,h,l,t*, where:

w, h are the width and height of the desired output area.

l, t are the offsets of the upper-left corner of the output area from the left (*l*) and from the top (*t*) of the maximum output area.

These four values must be given as percentages of the maximum output dimensions. The maximum output area is the area within the device clip limits.

FIT=S | l,t

The *FIT* parameter determines which part of the picture is to be printed. You can request the whole of the picture, scaled to fit the output area; or you can position the picture (actual size) anywhere within the output area. This could mean that the picture is clipped at the boundaries of the output area.

The default value of *FIT=S* causes the output to be scaled until the larger of the height or width just fits within the defined output area. The aspect ratio of the picture is maintained.

To print the picture in actual size, use *FIT=l,t*, where *l,t* are the coordinates of the point in the picture that you want positioned at the center of the output area: *l* is measured from the left edge of the picture; and *t* is measured from the top edge. The coordinates must be given as percentages of the actual dimensions of the picture.

XFM=0 | 1

The *XFM* parameter enables you to override the picture-positioning and clipping instructions that are provided by the *ARE* and *FIT* parameters, including their defaults.

The default value of *XFM=1* allows the appearance of the output to be determined by the settings of the *ARE* and *FIT* parameters.

A value of *XFM=0* yields output as specified in the picture file. For example, applications that use many different forms can define different positions on each form for their output.

COL=M | C

The *COL* parameter enables you to specify color output if you have a color printer.

A value of *COL=M* creates monochrome output (black foreground with no background color). This is supported by all devices.

A value of *COL=C* creates color output. If you request color output on a monochrome device, the printer presentation driver tries to satisfy your request, which can cause problems because the only color available is black. For example, if the picture file specifies a red line on a blue background, both are drawn in black.

The default is *COL=M* when you are addressing a monochrome printer and *COL=C* when you are addressing a color printer.

MAP=N | A

The *MAP* parameter enables you to decide how the *neutral* colors (those that are not specified in the picture file) are printed.

The default value of *MAP=N* yields a *normal* representation of the screen picture on a printed page, which means that the page background is white and the foreground is black.

A value of *MAP=A* provides the reverse of the normal representation: the background is black and the foreground is white on the printed page.

CDP=codepage

The *CDP* parameter overrides the code page to being used for PM_Q_RAW print jobs. The print queue driver uses DEVESC_SETMODE to set the code page, but not all printer drivers support this device escape.

XLT=0 | 1

The *XLT* parameter can eliminate the translation component when printing a metafile if *XLT=1*.

When the resolution of the device is higher than that of the world coordinate space, a small translation of world coordinate point (0,0) occurs on the device to preserve the accuracy of the mapping from world to device coordinate units. For example, (0,0) becomes (1,1) if there are 3 pels to every world coordinate.

Normally, this is not noticeable, but it can be a problem with some devices. For example, in order to draw a complete row of 80 characters using a device font, a device may require the text to start at device coordinate position zero. Starting at a position other than zero may cause one or more characters at the end of the row to be clipped. In such cases, elimination of the translation is important and can be accomplished by specifying *XLT=1*.

The default is *XLT=0* .

DEVOPENSTRUC Field - pszSpoolerParams

pszSpoolerParams (PSZ)

Spooler parameters.

Spooler parameters (optional) are separated by spaces. They are used for scheduling print jobs and are as follows:

- The form names that identify the paper to be used, for example, *FORM=A4 , A5 , ENV*. The form names are optional; but if they are provided, the spooler is able to hold off printing the jobs until the required form is installed in the printer. If the form name is not provided, the spooler attempts to print the job. The printer driver recognizes that there is a forms problem and displays a FORMS MISMATCH message box.
- Priority of the print job, for example, *PTY=60* . The priority is specified as an integer in the range 1 through 99; 99 is the highest. The default priority value is 50. The application can use the spooler priority parameter to prioritize its own jobs; however, it is not good practice for an application always to use priority 99 in an attempt to get its jobs printed first.

DEVOPENSTRUC Field - pszNetworkParams

pszNetworkParams (PSZ)

Network parameters.

Optional parameter that can be used to specify network options; for example, *USER=JOESMITH* .

DRAGITEM

Drag-object structure.

```
typedef struct _DRAGITEM {
    HWND      hwndItem;           /* Window handle of the source of the drag operation. */
    ULONG     ulItemID;           /* Information used by the source to identify the object being dragged. */
    HSTR      hstrType;           /* String handle of the object type. */
    HSTR      hstrRMF;            /* String handle of the rendering mechanism and format. */
    HSTR      hstrContainerName;  /* String handle of the name of the container holding the source object. */
    HSTR      hstrSourceName;     /* String handle of the name of the source object. */
    HSTR      hstrTargetName;     /* String handle of the suggested name of the object at the target. */
    SHORT     cxOffset;           /* X-offset from the pointer hot spot to the origin of the image that represents the object. */
    SHORT     cyOffset;           /* Y-offset from the pointer hot spot to the origin of the image that represents the object. */
    USHORT    fsControl;          /* Source-object control flags. */
    USHORT    fsSupportedOps;     /* Direct manipulation operations supported by the source object. */
} DRAGITEM;

typedef DRAGITEM *PDRAGITEM;
```

DRAGITEM Field - hwndItem

hwndItem ([HWND](#))

Window handle of the source of the drag operation.

DRAGITEM Field - ullItemID

ullItemID ([ULONG](#))

Information used by the source to identify the object being dragged.

DRAGITEM Field - hstrType

hstrType ([HSTR](#))

String handle of the object type.

The string handle must be created using the `DrgAddStrHandle` function. The string is of the form:

```
type[,type...]
```

The first type in the list must be the true type of the object. The following types are used by the OS/2* shell:

DRT_ASM	Assembler code
DRT_BASIC	BASIC code
DRT_BINDATA	Binary data
DRT_BITMAP	Bit map
DRT_C	C code
DRT_COBOL	COBOL code
DRT_DLL	Dynamic link library
DRT_DOSCMD	DOS command file
DRT_EXE	Executable file
DRT_FONT	Font
DRT_FORTRAN	FORTRAN code
DRT_ICON	Icon
DRT_LIB	Library
DRT_METAFILE	Metafile
DRT_OS2CMD	OS/2 command file
DRT_PASCAL	Pascal code
DRT_RESOURCE	Resource file
DRT_TEXT	Text
DRT_UNKNOWN	Unknown type.

DRAGITEM Field - hstrRMF

hstrRMF ([HSTR](#))

String handle of the rendering mechanism and format.

The string handle must be created using the `DrgAddStrHandle` function. The string is of the form:

```
mechfmt [, mechfmt ...]
```

where `mechfmt` can be in either of the following formats:

- `<mechanism(1),format(1)>`
- `(mechanism(1)[, mechanism(n)...]) x (format(1)[,format(n)...])`

The first mechanism/format pair must be the native rendering mechanism and format of the object.

Valid mechanisms are:

"DRM_DDE"	Dynamic data exchange
"DRM_OBJECT"	Item being dragged is a Workplace Shell object
"DRM_OS2FILE"	OS/2 file
"DRM_PRINT"	Object can be printed using direct manipulation.

Valid formats are:

"DRF_BITMAP"	OS/2 bit map
"DRF_DIB"	DIB
"DRF_DIF"	DIF
"DRF_DSPBITMAP"	Stream of bit-map bits
"DRF_METAFILE"	Metafile
"DRF_OEMTEXT"	OEM text
"DRF_OWNERDISPLAY"	Bit stream
"DRF_PTRPICT"	Printer picture
"DRF_RTF"	Rich text
"DRF_SYLK"	SYLK
"DRF_TEXT"	Null-terminated string
"DRF_TIFF"	TIFF
"DRF_UNKNOWN"	Unknown format.

DRAGITEM Field - hstrContainerName

hstrContainerName ([HSTR](#))

String handle of the name of the container holding the source object.

The string handle must be created using the `DrgAddStrHandle` function.

DRAGITEM Field - hstrSourceName

hstrSourceName ([HSTR](#))

String handle of the name of the source object.

The string handle must be created using the `DrgAddStrHandle` function.

DRAGITEM Field - hstrTargetName

hstrTargetName ([HSTR](#))

String handle of the suggested name of the object at the target.

It is the responsibility of the source of the drag operation to create this string handle before calling `DrgDrag`.

DRAGITEM Field - cxOffset

cxOffset ([SHORT](#))

X-offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from *cxOffset* in the [DRAGIMAGE](#) structure by `DrgDrag`.

DRAGITEM Field - cyOffset

cyOffset ([SHORT](#))

Y-offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from *cyOffset* in the [DRAGIMAGE](#) structure by `DrgDrag`.

DRAGITEM Field - fsControl

fsControl ([USHORT](#))

Source-object control flags.

DC_OPEN	Object is open
DC_REF	Reference to another object
DC_GROUP	Group of objects
DC_CONTAINER	Container of other objects

DC_PREPARE

Source requires a DM_RENDERPREPARE message before it establishes a data transfer conversation

DC_REMOVEABLEMEDIA

Object is on removable media, or object cannot be recovered after a move operation.

DRAGITEM Field - fsSupportedOps

fsSupportedOps (USHORT)

Direct manipulation operations supported by the source object.

DO_COPYABLE

Source supports DO_COPY

DO_LINKABLE

Source supports DO_LINK

DO_MOVEABLE

Source supports DO_MOVE.

DRAGIMAGE

Dragged-object-image structure which describes the images that are to be drawn under the direct-manipulation pointer for the duration of a drag operation.

```
typedef struct _DRAGIMAGE {
    USHORT    cb;           /* Size, in bytes, of the DRAGIMAGE structure. */
    USHORT    cptl;        /* The number of points in the point array if fl is specified as DRG_POLYGON. */
    LHANDLE    hImage;      /* Handle representing the image to display. */
    SIZE      sizlStretch;  /* Dimensions for stretching when fl is specified as DRG_STRETCH. */
    ULONG      fl;          /* Flags. */
    SHORT      cxOffset;    /* X-offset from the pointer hot spot to the origin of the image. */
    SHORT      cyOffset;    /* Y-offset from the pointer hot spot to the origin of the image. */
} DRAGIMAGE;

typedef DRAGIMAGE *PDRAGIMAGE;
```

DRAGIMAGE Field - cb

cb (USHORT)

Size, in bytes, of the DRAGIMAGE structure.

DRAGIMAGE Field - cptl

cptl (USHORT)

The number of points in the point array if fl is specified as DRG_POLYGON.

DRAGIMAGE Field - hImage

hImage ([LHANDLE](#))

Handle representing the image to display.

The type is determined by *fl* .

DRAGIMAGE Field - sizlStretch

sizlStretch ([SIZEL](#))

Dimensions for stretching when *fl* is specified as DRG_STRETCH.

DRAGIMAGE Field - fl

fl ([ULONG](#))

Flags.

DRG_ICON

hImage is an [HPOINTER](#).

DRG_BITMAP

hImage is an [HBITMAP](#).

DRG_POLYGON

hImage is a pointer to an array of points that will be connected with GpiPolyLine to form a polygon. The first point of the array should be (0,0), and the other points should be placed relative to this position.

DRG_STRETCH

If DRG_ICON or DRG_BITMAP is specified, the image is expanded or compressed to the dimensions specified by *sizlStretch* .

DRG_TRANSPARENT

If DRG_ICON is specified, an outline of the icon is generated and displayed instead of the original icon.

DRG_CLOSED

If DRG_POLYGON is specified, a closed polygon is formed by moving the current position to the last point in the array before calling GpiPolyLine.

DRAGIMAGE Field - cxOffset

cxOffset ([SHORT](#))

X-offset from the pointer hot spot to the origin of the image.

DRAGIMAGE Field - cyOffset

cyOffset ([SHORT](#))

Y-offset from the pointer hot spot to the origin of the image.

DRAGINFO

Drag-information structure.

```
typedef struct _DRAGINFO {
    ULONG        cbDraginfo; /* Structure size, in bytes. */
    USHORT       cbDragitem; /* Size, in bytes, of each DRAGITEM structure. */
    USHORT       usOperation; /* Modified drag operations. */
    HWND         hwndSource; /* Window handle of the source of the drag operation. */
    SHORT        xDrop; /* X-coordinate of drop point expressed in desktop coordinates. */
    SHORT        yDrop; /* Y-coordinate of drop point expressed in desktop coordinates. */
    USHORT       cditem; /* Count of DRAGITEM structures. */
    USHORT       usReserved; /* Reserved. */
} DRAGINFO;

typedef DRAGINFO *PDRAGINFO;
```

DRAGINFO Field - cbDraginfo

cbDraginfo (ULONG)
Structure size, in bytes.

The size includes the array of **DRAGITEM** structures.

DRAGINFO Field - cbDragitem

cbDragitem (USHORT)
Size, in bytes, of each **DRAGITEM** structure.

DRAGINFO Field - usOperation

usOperation (USHORT)
Modified drag operations.

An application can define its own modified drag operations for use when simulating a drop. These operations must have a value greater than DO_UNKNOWN. Possible values are described in the following list:

DO_DEFAULT	Execute the default drag operation. No modifier keys are pressed.
DO_COPY	Execute a copy operation. The Ctrl key is pressed.
DO_LINK	Execute a link operation. The Ctrl+Shift keys are pressed.
DO_MOVE	Execute a move operation. The Shift key is pressed.
DO_CREATE	Execute a create operation. (A template is being dropped.)
DO_NEW	Execute a create another operation. This value should be defined as DO_UNKNOWN+3 if it is not recognized in the current level of the toolkit.
DO_UNKNOWN	

An undefined combination of modifier keys is pressed.

DRAGINFO Field - hwndSource

hwndSource ([HWND](#))

Window handle of the source of the drag operation.

DRAGINFO Field - xDrop

xDrop ([SHORT](#))

X-coordinate of drop point expressed in desktop coordinates.

DRAGINFO Field - yDrop

yDrop ([SHORT](#))

Y-coordinate of drop point expressed in desktop coordinates.

DRAGINFO Field - cditem

cditem ([USHORT](#))

Count of [DRAGITEM](#) structures.

DRAGINFO Field - usReserved

usReserved ([USHORT](#))

Reserved.

Environment

SOM environment information.

```
typedef struct _Environment {  
    exception_type    _major;  
    struct exception {
```

```

    char                *_exception_name;
    void                *_params;
} exception;
char                *_somedAnchor;
} Environment;

```

This structure contains environmental information that can be passed between a caller and a called object when a method is executed. For example, it can be used to pass information about the user ID of a client or to return exception data to the client following a method call.

To set an exception value in the caller's Environment, a method implementation makes a call to the somSetException function; for example:

```

void somSetException ( Environment *env
                      exception_type major,
                      string exception_name,
                      void *params);

```

where *env* is a pointer to the Environment passed to the method, *_major* is an exception type, *_exception_name* is the exception name, and *_params* is a pointer to an initialized exception allocated by calling SOMMalloc.

Note: The somSetException method simply sets the exception value; it does not perform exit processing. If there are multiple calls to somSetException before the method returns, the call sees only the last exception value.

After a method returns, the calling client program can look at the Environment structure to determine whether an exception occurred. If *_major* does not equal NO_EXCEPTION, an exception was returned by the call. The user can retrieve the exception name and values by calling somExceptionID to retrieve a string containing the exception name and somExceptionValue to retrieve a pointer to the value of the exception. If NULL is passed as the Environment pointer in either of these calls, an implicit call is made to somGetGlobalEnvironment.

Environment Field - *_major*

_major (*exception_type*)

Type of exception returned from the call.

Possible values are described in the following list:

NO_EXCEPTION

An exception was not returned.

USER_EXCEPTION

A user exception was returned.

SYSTEM_EXCEPTION

A system exception was returned.

Environment Field - *_exception_name*

_exception_name (char *)

String containing the exception name.

Environment Field - *_params*

_params

Pointer to an initialized exception structure that must be allocated by SOMMalloc.

Environment Field - __somedAnchor

__somedAnchor (char *)

exception_type

An exception type.

```
enum exception_type {  
    NO_EXCEPTION;  
    USER_EXCEPTION;  
    SYSTEM_EXCEPTION;  
};
```

exception_type Field - NO_EXCEPTION

NO_EXCEPTION
An exception was not returned.

exception_type Field - USER_EXCEPTION

USER_EXCEPTION
A user exception was returned.

exception_type Field - SYSTEM_EXCEPTION

SYSTEM_EXCEPTION
A system exception was returned.

FIELDINFO

Structure that contains information about column data in the details view of the container control. The details view displays each FIELDINFO structure as a column of data that contains specific information about each container record. For example, one FIELDINFO structure, or column, might contain icons or bit maps that represent each container record. Another FIELDINFO structure might contain the date or time that each container record was created.

```

typedef struct _FIELDINFO {
    ULONG      cb;                /* Structure size. */
    ULONG      flData;            /* Data attributes. */
    ULONG      flTitle;           /* Column heading attributes. */
    PVOID      pTitleData;        /* Column heading data. */
    ULONG      offStruct;         /* Structure offset. */
    PVOID      pUserData;         /* Pointer to user data. */
    struct _FIELDINFO *pNextFieldInfo; /* Pointer to the next linked FIELDINFO data structure. */
    ULONG      cxWidth;           /* Column width. */
} FIELDINFO;

typedef FIELDINFO *PFIELDINFO;

```

FIELDINFO Field - cb

cb ([ULONG](#))

Structure size.

The size (in bytes) of the FIELDINFO structure.

FIELDINFO Field - flData

flData ([ULONG](#))

Data attributes.

Attributes of the data in a field.

- Specify one of the following for each column to choose the type of data that is displayed in each column:

CFA_BITMAPORICON

The column contains bit-map or icon data.

CFA_DATE

The data in the column is displayed in date format. National Language Support (NLS) is enabled for date format. Use the data structure described in [CDATE](#)

CFA_STRING

Character or text data is displayed in this column.

CFA_TIME

The data in the column is displayed in time format. National Language Support (NLS) is enabled for time format. Use the data structure described in [CTIME](#).

CFA_ULONG

Unsigned number data is displayed in this column. National Language Support (NLS) is enabled for number format.

- Specify any or all of the following column attributes:

CFA_FIREADONLY

Prevents text in a FIELDINFO data structure (text in a column) from being edited directly. This attribute applies only to columns for which the CFA_STRING attribute has been specified.

CFA_HORZSEPARATOR

A horizontal separator is provided beneath column headings.

CFA_INVISIBLE

Invisible container column. The default is visible.

CFA_OWNER

Ownerdraw is enabled for this container column.

CFA_SEPARATOR

A vertical separator is drawn after this column.

- Specify one of the following for each column to vertically position data in that column:

CFA_BOTTOM

Bottom-justifies field data.

CFA_TOP

Top-justifies field data.

CFA_VCENTER

Vertically centers field data. This is the default.

- Specify one of the following for each column to horizontally position data in that column. These attributes can be combined with the attributes used for vertical positioning of column data by using an OR operator (|).

CFA_CENTER

Horizontally centers field data.

CFA_LEFT

Left-justifies field data. This is the default.

CFA_RIGHT

Right-justifies field data.

FIELDINFO Field - fTitle

fTitle (ULONG)

Column heading attributes.

- Specify the following if icon or bit-map data is to be displayed in the column heading:

CFA_BITMAPORICON

The column heading contains icon or bit-map data. If CFA_BITMAPORICON is not specified, any data that is assigned to a column heading is assumed to be character or text data.

- Specify the following to prevent direct editing of a column heading:

CFA_FITTLEREADONLY

Prevents a column heading from being edited directly.

- Specify one of the following for each column heading to vertically position data in that column heading:

CFA_TOP

Top-justifies column headings.

CFA_BOTTOM

Bottom-justifies column headings.

CFA_VCENTER

Vertically centers column headings. This is the default.

- Specify one of the following for each column heading to horizontally position data in that column heading. These attributes can be combined with the attributes used for vertical positioning of column heading data by using an OR operator (|).

CFA_CENTER

Horizontally centers column headings.

CFA_LEFT

Left-justifies column headings. This is the default.

CFA_RIGHT

Right-justifies column headings.

FIELDINFO Field - pTitleData

pTitleData (PVOID)

Column heading data.

Column heading data, which can be a text string or an icon or bit map. The default is a text string. If the *flTitle* field is set to the CFA_BITMAPORICON attribute, this must be an icon or bit map.

FIELDINFO Field - offStruct

offStruct (ULONG)

Structure offset.

Offset from the beginning of a RECORDCORE structure to the data that is displayed in this column.

Note: If the CCS_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

FIELDINFO Field - pUserData

pUserData (PVOID)

Pointer to user data.

FIELDINFO Field - pNextFieldInfo

pNextFieldInfo (struct _FIELDINFO *)

Pointer to the next linked FIELDINFO data structure.

FIELDINFO Field - cxWidth

cxWidth (ULONG)

Column width.

Used to specify the width of a column. The default is an automatically sized column that is always the width of its widest element. If this field is set and the data is too wide, the data is truncated.

Handle

Address of a pointer to a char.


```
typedef Ptr *Handle;
```

HBITMAP

Bit-map handle.

```
typedef LHANDLE HBITMAP;
```

HMODULE

Module handle.

```
typedef LHANDLE HMODULE;
```

HPOINTER

Pointer handle.

```
typedef LHANDLE HPOINTER;
```

HPS

Presentation-space handle.

```
typedef LHANDLE HPS;
```

HRGN

Region handle.

```
typedef LHANDLE HRGN;
```

HSTR

String handle.

```
typedef LHANDLE HSTR;
```

HWND

Window handle.

```
typedef LHANDLE HWND;
```

ICONINFO

Icon information data structure.

```
typedef struct _ICONINFO {  
    ULONG      cb;           /* Length of the ICONINFO structure. */  
    ULONG      fFormat;      /* Indicates where the icon resides. */  
    PSZ        pszFileName;  /* Name of the file containing icon data. */  
    HMODULE     hmod;         /* Module containing the icon resource. */  
    ULONG      resid;         /* Identity of the icon resource. */  
    ULONG      cbIconData;    /* Length of the icon data in bytes. */  
    PVOID       pIconData;    /* Pointer to the buffer containing icon data. */  
} ICONINFO;  
  
typedef ICONINFO *PICONINFO;
```

ICONINFO Field - cb

cb ([ULONG](#))
Length of the ICONINFO structure.

ICONINFO Field - fFormat

fFormat ([ULONG](#))
Indicates where the icon resides.

Possible values are:

ICON_FILE	Icon file supplied.
ICON_RESOURCE	Icon resource supplied.
ICON_DATA	Icon data supplied.
ICON_CLEAR	

Go back to default icon.

ICONINFO Field - pszFileName

pszFileName ([PSZ](#))

Name of the file containing icon data.

This value is ignored if *fFormat* is not equal to ICON_FILE.

ICONINFO Field - hmod

hmod ([HMODULE](#))

Module containing the icon resource.

This value is ignored if *fFormat* is not equal to ICON_RESOURCE.

ICONINFO Field - resid

resid ([ULONG](#))

Identity of the icon resource.

This value is ignored if *fFormat* is not equal to ICON_RESOURCE.

ICONINFO Field - cblconData

cblconData ([ULONG](#))

Length of the icon data in bytes.

This value is ignored if *fFormat* is not equal to ICON_DATA.

ICONINFO Field - plconData

plconData ([PVOID](#))

Pointer to the buffer containing icon data.

This value is ignored if *fFormat* is not equal to ICON_DATA.

ISOString

ISO string.

```
typedef string ISOString;
```

LHANDLE

The handle of a resource.

```
typedef unsigned long LHANDLE;
```

LONG

Signed integer in the range -2 147 483 648 through 2 147 483 647.

```
#define LONG long
```

Note: Where this data type represents a graphic coordinate in world or model space, its value is restricted to -134 217 728 through 134 217 727.

A graphic coordinate in device or screen coordinates is restricted to -32 768 through 32 767.

The value of a graphic coordinate may be further restricted by any transforms currently in force, including the positioning of the origin of the window on the screen. In particular, coordinates in world or model space must not generate coordinate values after transformation (that is, in device or screen space) outside the range -32 768 through 32 767.

MATRIXLF

Matrix-elements structure.

```
typedef struct _MATRIXLF {  
    FIXED    fxM11; /* First element of first row. */  
    FIXED    fxM12; /* Second element of first row. */  
    LONG     lM13; /* Third element of first row. */  
    FIXED    fxM21; /* First element of second row. */  
    FIXED    fxM22; /* Second element of second row. */  
    LONG     lM23; /* Third element of second row. */  
    LONG     lM31; /* First element of third row. */  
    LONG     lM32; /* Second element of third row. */  
    LONG     lM33; /* Third element of third row. */  
} MATRIXLF;  
  
typedef MATRIXLF *PMATRIXLF;
```

MATRIXLF Field - fxM11

fxM11 (FIXED)
First element of first row.

MATRIXLF Field - fxM12

fxM12 (FIXED)
Second element of first row.

MATRIXLF Field - IM13

IM13 (LONG)
Third element of first row.

MATRIXLF Field - fxM21

fxM21 (FIXED)
First element of second row.

MATRIXLF Field - fxM22

fxM22 (FIXED)
Second element of second row.

MATRIXLF Field - IM23

IM23 (LONG)
Third element of second row.

MATRIXLF Field - IM31

IM31 ([LONG](#))

First element of third row.

MATRIXLF Field - IM32

IM32 ([LONG](#))

Second element of third row.

MATRIXLF Field - IM33

IM33 ([LONG](#))

Third element of third row.

MENUITEM

Menu item.

```
typedef struct _MENUITEM {  
    SHORT      iPosition;    /* Position. */  
    USHORT     afStyle;      /* Style. */  
    USHORT     afAttribute;  /* Attribute. */  
    USHORT     id;           /* Identity. */  
    HWND       hwndSubMenu;  /* Submenu. */  
    ULONG      hItem;        /* Item. */  
} MENUITEM;  
  
typedef MENUITEM *PMENUITEM;
```

MENUITEM Field - iPosition

iPosition ([SHORT](#))

Position.

MENUITEM Field - afStyle

afStyle ([USHORT](#))

Style.

MENUITEM Field - afAttribute

afAttribute (**USHORT**)
Attribute.

MENUITEM Field - id

id (**USHORT**)
Identity.

MENUITEM Field - hwndSubMenu

hwndSubMenu (**HWND**)
Submenu.

MENUITEM Field - hItem

hItem (**ULONG**)
Item.

MPARAM

A 4-byte message-dependent parameter structure.

```
typedef VOID *MPARAM;
```

Certain elements of information, placed into the parameters of a message, have data types that do not use all four bytes of this data type. The rules governing these cases are:

BOOL	The value is contained in the low word and the high word is 0.
SHORT	The value is contained in the low word and its sign is extended into the high word.
USHORT	The value is contained in the low word and the high word is 0.
NULL	The entire four bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

MRESULT

A 4-byte message-dependent reply parameter structure.

```
typedef VOID *MRESULT;
```

Certain elements of information, placed into the parameters of a message, have data types that do not use all four bytes of this data type. The rules governing these cases are:

BOOL	The value is contained in the low word and the high word is 0.
SHORT	The value is contained in the low word and its sign is extended into the high word.
USHORT	The value is contained in the low word and the high word is 0.
NULL	The entire four bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

ODACCEL

Accelerator information.

```
typedef struct _ODACCEL {  
    ACCEL    aAccel[MAX];  
} ODACCEL;
```

ODACCEL Field - aAccel[MAX]

aAccel[MAX] (ACCEL)
An array of ACCEL structures containing accelerator information.

ODActionData

A structure containing the action data for **Undo** and **Redo** commands.

```
typedef ODByteArray ODActionData;
```

ODActionType

An enumeration specifying the possible values for an Undo action.

```
enum ODActionType {  
    kODBeginAction;  
    kODEndAction;  
    kODSingleAction;  
};
```

ODActionType Field - kODBeginAction

kODBeginAction

The action was the first of a two-part action (such as the drag part of a drag and drop action).

ODActionType Field - kODEndAction

kODEndAction

The action was the second of a two-part action (such as the drop part of a drag and drop action).

ODActionType Field - kODSingleAction

kODSingleAction

The action was a single action.

ODBoolean

A Boolean value; the size of this type is platform-dependent.

```
typedef boolean ODBoolean;
```

kODTrue	True
kODFalse	False

ODByteArray

This data type represents foreign data types larger than 4 bytes, or variable-length data in general.

```
typedef _IDL_SEQUENCE_octet ODByteArray;
```

In most cases, this structure is used to pass raw data between OpenDoc and its clients. If a method expects a specific structure in the *_buffer* field, the method description explains what structure this field should contain.

ODClipboardKey

Clipboard key.

```
typedef ODULong ODClipboardKey;
```

ODCloneKind

An enumeration specifying the possible semantic values of a clone operation.

```
typedef ODULong ODCloneKind;
```

Values of this type are passed to the draft object's [BeginClone](#) method.

This type can be set to one of the following values:

kODCloneCopy	Copy into the content storage unit of the clipboard object or the drag-and-drop object.
kODCloneCut	Cut into the content storage unit of the clipboard object or the drag-and-drop object.
kODCloneDropCopy	Copy at the destination of a drop.
kODCloneDropMove	Move at the destination of a drop.
kODCloneFromLink	Clone from a link.
kODClonePaste	Paste from the content storage unit of the clipboard object or the drag-and-drop object.
kODCloneToLink	Clone to a link.

ODCloneKindConsts

An enumeration used to indicate the desired semantics of a clone operation (a value for [ODCloneKind](#)).

```
enum ODCloneKindConsts {  
    kODCloneCopy;  
    kODCloneCut;  
    kODClonePaste;  
    kODCloneDropCopy;  
    kODCloneDropMove;  
    kODCloneAll;  
};
```

ODCloneKindConsts Field - kODCloneCopy

kODCloneCopy
Copy an object from the source to the data-transfer object.

ODCloneKindConsts Field - kODCloneCut

kODCloneCut
Clone an object from the source to the data-transfer object.

ODCloneKindConsts Field - kODClonePaste

kODClonePaste

Paste an object from the data-transfer object to the destination of a drop.

ODCloneKindConsts Field - kODCloneDropCopy

kODCloneDropCopy

Copy an object to the destination of a drop.

ODCloneKindConsts Field - kODCloneDropMove

kODCloneDropMove

Move an object to the destination of a drop.

ODCloneKindConsts Field - kODCloneAll

kODCloneAll

Copy all objects from the source to the data-transfer object.

ODCodePage

Platform-dependent code page.

```
typedef unsigned long ODCodePage;
```

ODCommandID

A 32-bit value representing the Command ID associated with a particular menu/item combination.

```
typedef ODSLong ODCommandID;
```

ODContainerID

A structure identifying a container.

```
typedef ODBinaryArray ODContainerID;
```

The `_buffer` field of this byte array holds a container-suite-specific identifier for a container.

The structure of the `_buffer` field depends on the type of container. For example, the identifier for a Bento file container specifies a file-system file; the identifier for a Bento memory container is a handle for a relocatable memory block.

ODContainerName

A user-readable name for a container object.

```
typedef ODName ODContainerName;
```

ODContainerSuite

An opaque type representing a specific container suite.

```
typedef ODISOStr ODContainerSuite;
```

ODContainerType

A string used to specify a type of storage container.

```
typedef ODISOStr ODContainerType;
```

On each platform, OpenDoc has a default container type that it uses for documents and a default container type that it used for the drag and drop and the clipboard.

This type can be set to one of the following values:

kODBentoFileContainer	The Bento container class for documents.
kODBentoMemoryContainer	The Bento container class for drag and drop and the clipboard.
kODDefaultFileContainer	The default container type for documents on this platform.
kODDefaultMemoryContainer	The default container type for drag and drop and the clipboard on this platform.

ODContour

Contour of a polygon. A structure of type [ODBinaryArray](#) representing a contour of a polygon.

```
typedef ODByteArray ODContour;
```

A contour consists of a closed loop of three or more points connected by straight edges. The `_buffer` field of a contour byte array points to a 32-bit signed value, indicating the number of points in the contour, followed by the specified number of [ODPoint](#) structures, representing the individual points.

The order of points in a contour is significant: for a right-oriented coordinate system like GPI, points arranged in a counter-clockwise order represent a positive area, while in the opposite order they represent a negative area or hole (in a left-oriented coordinate system such as QuickDraw, the reverse is true).

An ODContour structure exists only as a component of an [ODPolygon](#) structure (described next).

ODCoordinate

A 32-bit value representing a spatial coordinate in a document or window.

```
typedef ODFixed ODCoordinate;
```

By default, OpenDoc represents a coordinate with 16 integer bits and 16 fractional bits, which is identical to a value of the [ODFixed](#) type. You can partition the bits of an ODCoordinate in any way, provided that you shift the values appropriately when passing information outside of OpenDoc. (If you are using a graphics system that handles arbitrary transformations, you can do this automatically by assigning a scaling factor to your internal transform.)

ODDescType

An OSA events descriptor type. This is a wrapper for the OSA events type DescType.

```
typedef ODULong ODDescType;
```

You should use these constants only when writing to an **aete** resource that overrides some of OpenDoc's standard scripting support.

General Descriptor Types

The following descriptor types correspond to OSA events object model properties that represent general descriptor types.

kAEOpenDocSuite	The type code for the OpenDoc suite in the aete resource (value odst).
kODStandardPartTokenType	The type code for the standard part token, as returned from the name resolver object's Resolve method. This constant is equivalent to the cPart constant.

OpenDoc-Suite Classes

The following descriptor types correspond to OSA events object model properties that represent general elements (classes) of the OpenDoc suite.

cDraft	A draft (value drft).
cPart	A part (value part).
clconFamily	An icon family (value ifam).

Part-Information Properties

The following descriptor types correspond to OSA events object model properties that represent the properties shown in the Part Info dialog window.

pASCreationDate	The part's creation date (value ascd).
pASModificationDate	The part's modification date (value asmo).
pAuthor	The part's author (value auth).
pBundled	Specifies whether the part is bundled (value bndl).

pCategory	The part's category (value pcat).
pComment	A comment about the part (value comt).
pContainer	The part that contains this part (value ctnr).
pEditor	The part's editor (value edtr).
pEditorName	The part's name (value enam).
pIcon	The part's icon (value iimg).
pKind	The part kind of the part (value kind).
pShowLinks	Specifies that all part editors should display all link borders in all windows displaying the document (value slnk).
pSize	The part's size (value size).
pStationery	Specifies that the part is a stationery part (value stat).
pUniqueID	The ID number for the part (value ID).
pViewType	The part's view type (value vwty).
enumViewType	The part's view type (value vwty). This constant is equivalent to the pViewType constant.

View types

The following descriptor types represent possible view types for a part, as shown in the Part Info dialog window.

kAEODFrame	Framed view type.
kAEODLargeIcon	Large (standard)-icon view type.
kAEODSmallIcon	Small-icon view type.
kAEODThumbnail	Thumbnail icon view type.

ODDocumentID

An identifier for a document.

```
typedef ODID ODDocumentID;
```

kODDefaultDocument	The default document ID.
kODNULLID	A null document ID.

ODDocumentName

A user-readable name for a document.

```
typedef ODName ODDocumentName;
```

ODDoneState

An enumeration indicating the state of an undo action.

```
enum ODDoneState {  
    kODDone;  
    kODRedone;  
    kODUndone;  
};
```

ODDoneState Field - kODDone

kODDone

The action was done and is now on the Undo stack.

ODDoneState Field - kODRedone

kODRedone

The action was done, undone, and redone and is now back on the Undo stack.

ODDoneState Field - kODUndone

kODUndone

The action was done and undone and is now on the Redo stack.

ODDraftID

A 32-bit value used as an identifier for a draft.

```
typedef ODID ODDraftID;
```

ODDraftKey

A 32-bit value used as a key for a cloning transaction.

```
typedef ODULong ODDraftKey;
```

ODDraftName

A name for a draft.

```
typedef ODISOStr ODDraftName;
```

ODDraftPermissions

An enumeration specifying the possible values for draft permissions.

```
typedef ODULong ODDraftPermissions;
```

kODExclusiveWrite	Read access and exclusive-write access.
kODNone	No access.
kODReadOnly	Read-only access.
kODSharedWrite	Read access and shared-write access.
kODTransient	Navigation-only access.

The Bento container suite supports only the kODReadOnly and kODExclusiveWrite draft permissions.

ODDragResult

A flag indicating whether this part can accept the dragged data in the given facet.

```
typedef ODBoolean ODDragResult;
```

ODDropResult

An enumeration used to specify the result of a drop operation.

```
enum ODDropResult {  
    kODDropCopy;  
    kODDropFail;  
    kODDropMove;  
    kODDropUnfinished;  
};
```

ODDropResult Field - kODDropCopy

kODDropCopy
Successful synchronous drop with copy semantics.

ODDropResult Field - kODDropFail

kODDropFail

Unsuccessful synchronous drop.

ODDropResult Field - kODDropMove

kODDropMove

Successful synchronous drop with move semantics.

ODDropResult Field - kODDropUnfinished

kODDropUnfinished

An asynchronous drop was started.

ODEditor

An opaque, platform-specific type identifying a specific part editor, or kODNoEditor for a null OpenDoc editor.

```
typedef ODISOStr ODEditor;
```

ODError

A 32-bit exception code.

```
typedef ODSLong ODError;
```

[Return Codes by Number](#) and [Return Codes by Name](#) describe the constants defined for this type.

ODEventClass

The event class of an OSA event. This is a wrapper for the OSA events type AEEEventClass.

```
typedef ODULong ODEventClass;
```

ODEventData

A platform-specific structure representing an event.

```
typedef struct _ODEventData {  
    HWND      hwnd;  
    ULONG     msg;  
    MPARAM    mp1;  
    MPARAM    mp2;  
    MRESULT   result;  
} ODEventData;  
  
typedef ODEventData *PODEventData;
```

ODEventData Field - hwnd

hwnd ([HWND](#))
Window handle that the event is associated with.

ODEventData Field - msg

msg ([ULONG](#))
Message ID.

ODEventData Field - mp1

mp1 ([MPARAM](#))
Message parameter 1.

ODEventData Field - mp2

mp2 ([MPARAM](#))
Message parameter 2.

ODEventData Field - result

result ([MRESULT](#))
Message return value.

ODEventID

The ID of an OSA event. This is a wrapper for the OSA events type AEEventID.

```
typedef ODULong ODEventID;
```

ODEventInfo (OS/2)

A structure containing context-specific event information.

```
typedef struct _ODEventInfo {  
    ODFrame      *embeddedFrame;  
    ODFacet      *embeddedFacet;  
    ODPoint      where;  
    ODULong      flags;  
} ODEventInfo;
```

ODEventInfo (OS/2) Field - embeddedFrame

embeddedFrame (ODFrame *)

The frame within which the event occurred. Only relevant for events that are delivered to the containing part.

ODEventInfo (OS/2) Field - embeddedFacet

embeddedFacet (ODFacet *)

The facet within which the event occurred. Only relevant for events that are delivered to the containing part.

ODEventInfo (OS/2) Field - where

where (ODPoint)

The position in local (frame) coordinates where the event occurred. Only relevant for mouse-related events.

ODEventInfo (OS/2) Field - flags

flags (ODULong)

A flag indicating where the event occurred. This parameter can be set to one of the following values:

kODPropagated	The event was propagated from an embedded part.
kODInBorder	The (mouse) event occurred over the part's active border.
kODInEmbedded	The (mouse) event occurred over an embedded part.

ODEventType

A 16-bit platform-specific code specifying an event type.

```
typedef ODUShort ODEventType;
```

ODException

A structure describing an exception.

```
typedef struct _ODException {  
    ODError    error;  
    char       message[256];  
} ODException;
```

ODException Field - error

error (ODError)

The error code identifying the exception.

ODException Field - message[256]

message[256] (char)

A string, used only for debugging purposes, that contains additional information about the exception.

ODFIELDINFO

```
typedef struct _ODFIELDINFO {  
    ODULong    cb;
```

```

    ODULong    flDataAttrs;
    ODULong    flTitleAttrs;
    STR64      pTitleText;
} ODFIELDINFO;

typedef ODFIELDINFO *PODFIELDINFO;

```

ODFIELDINFO Field - cb

cb ([ODULong](#))

The size (in bytes) of the [FIELDINFO](#) structure.

ODFIELDINFO Field - flDataAttrs

flDataAttrs ([ODULong](#))

The display attributes of the display data.

- Specify one of the following for each column to choose the type of data that is displayed in each column:
 - CFA_BITMAPORICON**
The column contains bit-map or icon data.
 - CFA_DATE**
The data in the column is displayed in date format. National Language Support (NLS) is enabled for date format. Use the data structure described in [CDATE](#).
 - CFA_STRING**
Character or text data is displayed in this column.
 - CFA_TIME**
The data in the column is displayed in time format. National Language Support (NLS) is enabled for time format. Use the data structure described in [CTIME](#).
 - CFA_ULONG**
Unsigned number data is displayed in this column. National Language Support (NLS) is enabled for number format.
- Specify any or all of the following column attributes:
 - CFA_FIREADONLY**
Prevents text in a [FIELDINFO](#) data structure (text in a column) from being edited directly. This attribute applies only to columns for which the CFA_STRING attribute has been specified.
 - CFA_HORZSEPARATOR**
A horizontal separator is provided beneath column headings.
 - CFA_INVISIBLE**
Invisible container column. The default is visible.
 - CFA_OWNER**
Ownerdraw is enabled for this container column.
 - CFA_SEPARATOR**
A vertical separator is drawn after this column.
- Specify one of the following for each column to vertically position data in that column:
 - CFA_BOTTOM**
Bottom-justifies field data.
 - CFA_TOP**

Top-justifies field data.

CFA_VCENTER

Vertically centers field data. This is the default.

- Specify one of the following for each column to horizontally position data in that column. These attributes can be combined with the attributes used for vertical positioning of column data by using an OR operator (|).

CFA_CENTER

Horizontally centers field data.

CFA_LEFT

Left-justifies field data. This is the default.

CFA_RIGHT

Right-justifies field data.

ODFIELDINFO Field - flTitleAttrs

flTitleAttrs (ODULong)

The display attributes of the column heading.

- Specify the following if icon or bit-map data is to be displayed in the column heading:

CFA_BITMAPORICON

The column heading contains icon or bit-map data. If CFA_BITMAPORICON is not specified, any data that is assigned to a column heading is assumed to be character or text data.

- Specify the following to prevent direct editing of a column heading:

CFA_FITTLEREADONLY

Prevents a column heading from being edited directly.

- Specify one of the following for each column heading to vertically position data in that column heading:

CFA_TOP

Top-justifies column headings.

CFA_BOTTOM

Bottom-justifies column headings.

CFA_VCENTER

Vertically centers column headings. This is the default.

- Specify one of the following for each column heading to horizontally position data in that column heading. These attributes can be combined with the attributes used for vertical positioning of column heading data by using an OR operator (|).

CFA_CENTER

Horizontally centers column headings.

CFA_LEFT

Left-justifies column headings. This is the default.

CFA_RIGHT

Right-justifies column headings.

ODFIELDINFO Field - pTitleText

pTitleText (STR64)

The column heading text.

ODFileRefNum

A 32-bit file reference number.

```
typedef ODForeign ODFileRefNum;
```

ODFileSpec

A 32-bit structure specifying a file on disk.

```
typedef ODForeign ODFileSpec;
```

ODFixed

A 32-bit fixed-point value used to represent non-integer numbers in the range [-32768, 32768).

```
typedef ODSLong ODFixed;
```

The high 16 bits (including a sign bit) represent the integer part, and the low 16 bits represent a fractional part. In effect, the *binary point* is in the middle of the number.

You can convert an integer to ODFixed by shifting it left 16 bits. You can round an ODFixed value to an integer by adding 0x8000 (0.5) and shifting the result right 16 bits. You can convert between ODFixed and floating-point by multiplying or dividing by 65536.0. You can add and subtract ODFixed values as though they were integers; however, you cannot multiply or divide them directly.

ODFlags

An unsigned 32-bit value used to represent a collection of flags.

```
typedef ODULong ODFlags;
```

ODFloat

A floating-point value. The size of this type is platform-dependent.

```
typedef float ODFloat;
```

ODFocusType

A string of type [ODType](#) used to identify a focus type.

```
typedef OType ODFocusType;
```

A string of this type can be tokenized using the session object's [Tokenize](#) method. Arbitrator methods use the tokenized forms of these strings.

This type can take one of the following values:

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as PageUp). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events (Shift +click and Ctrl +click). OpenDoc draws the active frame border around all facets of this frame.

ODForeign

A 32-bit platform-dependent type.

```
typedef void *ODForeign;
```

ODFract

A 32-bit fixed-point value used to represent non-integer numbers in the range [-2, 2).

```
typedef ODSLong ODFract;
```

The high two bits (including the sign bit) represent the integer part and the low 30 bits represent a fractional part.

You can convert an integer to ODFract by shifting it left 30 bits. You can round an ODFract value to an integer by adding 0x20000000 (.5) and shifting the result right 30 bits. You can convert between ODFract and floating-point numbers by multiplying or dividing by a scaling factor of 1073741824.0. You can add and subtract ODFract values as though they were integers; however, you cannot multiply or divide them directly.

ODFramePosition

An enumeration specifying the possible positions of a frame relative to a sibling frame.

```
enum ODFramePosition {  
    kODFrameBehind;  
    kODFrameInFront;  
};
```

ODFramePosition Field - kODFrameBehind

kODFrameBehind

This frame is behind its sibling.

ODFramePosition Field - kODFrameInFront

kODFrameInFront

This frame is in front of its sibling.

ODGeometryMode

An enumeration specifying the geometry modes of a shape object, indicating whether the shape is required to maintain its geometric (polygonal) representation.

```
enum ODGeometryMode {  
    kODLoseGeometry;  
    kODNeedsGeometry;  
    kODPreserveGeometry;  
};
```

This datatype tells whether the shape's geometric information (its polygonal representation) will be needed in the future.

ODGeometryMode Field - kODLoseGeometry

kODLoseGeometry

The shape does not need to use a polygon to describe its geometric representation. The polygonal representation can be discarded in order to optimize speed, at the expense of accuracy and persistent storage capability. A facet's clip shape will generally have this mode.

ODGeometryMode Field - kODNeedsGeometry

kODNeedsGeometry

The shape must maintain its polygonal representation. A facet's frame shape and used shape will have this mode because they are stored persistently in polygonal form.

ODGeometryMode Field - kODPreserveGeometry

kODPreserveGeometry

The shape must maintain its polygonal representation for as long as possible. This is the default value.

ODGraphicsSystem

A 16-bit value specifying a native platform graphics system.

```
typedef ODSShort ODGraphicsSystem;
```

This data type is used to identify the type of graphic system on platforms with more than one graphic system.

Every graphics system supported by OpenDoc should be identified by a unique ODGraphicsSystem value. These values are used by transform objects and shape objects to tag graphics-system-specific data.

In OS/2, parameters of this type should always be defined as kODGPI.

ODHandle

A 32-bit value representing a handle.

```
typedef ODULong ODHandle;
```

ODHighlight

An enumeration specifying the possible highlight states of a facet.

```
enum ODHighlight {  
    kODDimHighlight;  
    kODFullHighlight;  
    kODNoHighlight;  
};
```

ODHighlight Field - kODDimHighlight

kODDimHighlight

The facet is highlighted in background style.

ODHighlight Field - kODFullHighlight

kODFullHighlight
The facet is highlighted in foreground style.

ODHighlight Field - kODNoHighlight

kODNoHighlight
The facet is not highlighted.

ODIconFamily

An opaque platform-specific type that contains a collection of icons for rendering a part or other content in iconic form.

```
typedef ODHandle ODIconFamily;
```

kODLargeIconSize	The size of a large icon, which is 32x32 pixels.
kODSmallIconSize	The size of a small icon, which is 16x16 pixels.
kODThumbnailSize	The size of a thumb nail icon, which is 64x64 pixels.
kODTinyIconSize	The size of a tiny icon, which is 12x12 pixels.

ODID

A 32-bit value used as an identifier for a particular type of object (for example, a document or a storage unit).

```
typedef ODULong ODID;
```

This type can be set to one of the following values:

kODIDAll	Any ID. This value is used when specifying which storage units (or other objects) are of interest to a particular operation.
kODIDWild	Any ID. This value is used when specifying which storage units (or other objects) are of interest to a particular operation.
kODNULLID	A null ID or an invalid ID.

Note that these three constants have the same value; their different names provide clarity to the code that uses them. Typically kODNULLID is used to mean an absent or invalid ID. kODIDAll or kODIDWild are used to specify no restrictions on the scope of a cloning operation.

ODIdleFrequency

An 32-bit unsigned number representing the frequency, in ticks (60ths of a second), at which null events are sent during idle times.

```
typedef ODULong ODIdleFrequency;
```

ODInfoType

An opaque type for the part info data.

```
typedef ODULong ODInfoType;
```

To use the data, you should cast an ODInfoType value to and from your part's own representation of the data.

ODISOStr

A pointer to an ISO string.

```
typedef string ODISOStr;
```

An ISO string is a string composed of ASCII characters, terminated by a null character (zero byte). Because the first null character terminates the string, null characters cannot be embedded.

Name-Mapping Resources

The following constants identify the name-mapping (**nmap**) resources that a part editor uses to describe the kinds of data it can edit. OpenDoc uses these resources to construct the specified tables.

kODCategoryUserString	A table that maps a part category to a user-visible string naming that category.
kODContainerSuite	A table that maps a container type to the container suite ID for that type.
kODEditorHelpFile	A table that maps an editor ID to the name of that editor's help file.
kODEditorKinds	A table that map an editor ID to the part kinds that the editor supports.
kODEditorUserString	A table the maps an editor ID to a user-visible string naming that editor.
kODKind	A table that maps a part kind to the categories that the kind belongs to.
kODKindUserString	A table the maps a part kind to a user-visible string naming that kind.
kODViewer	A table that maps a part viewer ID to the viewer type. Currently, the only supported viewer type is kODSimpleViewer.

Part Categories

The following constants are used to identify part categories in the part registration.

kODCategoryCalendar	Calendar data.
kODCategoryChart	Chart data.
kODCategoryCompressed	Compressed data (such as in ZIP format)
kODCategoryConnection	Connection part kinds.
kODCategoryControl	Controls, such as a button.
kODCategoryControlPanel	Control panels.
kODCategoryDatabase	Databases, such as in DB/2 format.
kODCategoryDrawing	Drawing part kinds, such as in PM metafile format.
kODCategoryExecutable	Executable part kinds, such as in .COM, .EXE, or other executable format.
kODCategoryForm	Form data.
kODCategoryFormula	Formula data, such as in an equation-editor format.

kODCategoryKey	Key data.
kODCategoryLocator	Locator data, such as a URL.
kODCategoryMailingLabel	Mailing labels.
kODCategoryMovie	Movie data, such as in AVI format.
kODCategoryOutline	Document-outline data.
kODCategoryOS2StdTypes	A generic category for the standard OS/2 types.
kODCategoryOS2TypesDisplayName	The display name for kODCategoryOS2StdTypes
kODCategoryPageLayout	Page-layout data.
kODCategoryPainting	Painting data, such as in TIFF format.
kODCategoryPersonalInfo	Personal information, such as on business cards.
kODCategoryPlainText	Plain text, such as in ASCII code page 850.
kODCategoryPresentation	Presentation data.
kODCategoryPrinter	Printer data.
kODCategoryProject	Project-management data, such as in GA SuperProject format.
kODCategoryQuery	Database queries, such as in SQL format.
kODCategorySampledSound	Sampled sounds, such as in the WOW format.
kODCategoryScript	Scripting data, such as in Object REXX or any other OSA-compliant scripting-language format.
kODCategorySignature	Digital-signature data.
kODCategorySpace	Space data, such as a folder, hard disk, or server.
kODCategorySpreadsheet	Spreadsheet data.
kODCategoryStructuredSound	Structured sound, such as in MIDI format.
kODCategoryStyledText	Styled text, such as in RTF format.
kODCategory3DGraphic	3D graphics part kinds.
kODCategoryTable	Tables, such as in tab-delimited text format.
kODCategoryTime	Part kinds related to time, for example, clocks.
kODCategoryUtility	Utilities.

Part Editors and Viewers

The following constants are used to identify part editors and viewers.

kODBlackBoxHandlerOfLastResort	The editor ID for the editor of the last resource used for any part for which a suitable editor cannot be found.
kODSimpleViewer	A viewer for a simple part.

Data Types in Resources

The following constants identify the various data types.

kODIsAnISOStringID	An ISO string.
kODIsAnISOStringListID	A list of ISO strings.
kODIsINTLTextID	International text structure.
kODIsPltfmTypeSpaceID	A platform type space (file or data).
kODIsHelpFileNameID	A help file name.

Standard ISO Prefixes

The following constant is used to identify standard ISO prefixes.

Data Interoperability

The following constants are used to identify standard presentation manager (PM) formats and drag-and-drop rendering mechanism and format pairs.

kODKindAudioAU	AU Waveform Audio
kODKindGraphicsCGM	CGM graphics
kODKindImageGIF	GIF image
kODKindImageJPEG	JPEG image
kODKindMusicMIDI	MIDI
kODKindPlainText	Plain ASCII text in code page 850
kODKindTextRTF10	Rich Text Format 1.0
kODKindVideoMPEG	MPEG video

ODIText

A platform-specific structure representing a user-visible, international text string.

```
typedef struct _ODIText {
    ODITextFormat    format;
    ODByteArray      text;
} ODIText;
```

The characters in an international text string are represented by 8-bit bytes, thus a total 256 byte values can be used; in contrast, the only the 128 ASCII characters can be used in an ISO string.

The kODISO10646_1993BaseEncoding constant is an 18-bit code indicating the document interchange format of text written out by OpenDoc. This format corresponds to ISO standard 10646-1, 1993, fully decomposed; it is the only interchange format for text currently supported by OpenDoc.

When international text is stored in an interchange format, the interchange-format code is stored with the encoded text.

ODIText Field - format

format (ODITextFormat)

The format of the text. This parameter can be set to the following value:

kODISO10646_1993BaseEncoding	The unicode format for IText data in storage units.
kODOS2PlainText	The international text standard of language code/language code/string.

ODIText Field - text

text (ODByteArray)

The text string, represented as an ODByteArray structure in the specified format.

In the kODOS2PlainText format, the *_buffer* field of this byte array contains two 16-bit values followed by the raw text. The first 16-bit value is the script code, the second is the language code. The *_length* field of this byte array indicates the entire length of the *_buffer* field in 8-bit bytes; subtract the length of the two codes (4 bytes) to get the text length.

ODITextFormat

A 32-bit value specifying a text format.

```
typedef long ODITextFormat;
```

kODISO10646_1993BaseEncoding The unicode format for IText data in storage units.

kODOS2PlainText The international text standard of language code/language code/string.

ODLangCode

Platform-dependent language code.

```
typedef unsigned long ODLangCode;
```

ODLinkDescription

A string used to hold the user-given name to a link or link source.

```
typedef ODIText ODLinkDescription;
```

ODLinkID

A 32-bit value representing a link or link source.

```
typedef ODID ODLinkID;
```

ODLinkInfo

A structure that contains information about a link destination for display in the Link Destination Info dialog box.

```
typedef struct _ODLinkInfo {  
    ODType      kind;  
    ODTime      creationTime;  
    ODTime      changeTime;  
    ODUpdateID  change;  
    ODBoolean    autoUpdate;  
} ODLinkInfo;
```

ODLinkInfo Field - kind

kind (ODType)

The part kind used by the destination of the link.

ODLinkInfo Field - creationTime

creationTime (ODTime)

The date and time of the creation of this link destination.

ODLinkInfo Field - changeTime

changeTime (ODTime)

The date and time of the latest source change read by this destination.

ODLinkInfo Field - change

change (ODUpdateID)

The update ID of the source update last read by this destination.

ODLinkInfo Field - autoUpdate

autoUpdate (ODBoolean)

A flag indicating whether this destination updates automatically.

kODTrue

This destination updates automatically.

kODFalse

This destination does not update automatically.

ODLinkInfoAction

An enumeration specifying the kind of action to be taken as the result of user selections in either the Link Source Info dialog box or the Link Destination Info dialog box.

```
enum ODLinkInfoAction {
```



```
kODLinkInfoBreakLink;  
kODLinkInfoCancel;  
kODLinkInfoFindSource;  
kODLinkInfoOK;  
kODLinkInfoUpdateNow;  
};
```

ODLinkInfoAction Field - kODLinkInfoBreakLink

kODLinkInfoBreakLink
Break the link.

ODLinkInfoAction Field - kODLinkInfoCancel

kODLinkInfoCancel
Take no action; the user cancelled the dialog box.

ODLinkInfoAction Field - kODLinkInfoFindSource

kODLinkInfoFindSource
Display the source of the link. Relevant only in the Link Destination Info dialog box.

ODLinkInfoAction Field - kODLinkInfoOK

kODLinkInfoOK
Accept any new settings selected by the user in the dialog box.

ODLinkInfoAction Field - kODLinkInfoUpdateNow

kODLinkInfoUpdateNow
Update the link immediately.

ODLinkInfoResult

A structure that contains the results of user selections in either the Link Source Info dialog box or the Link Destination Info dialog box.

```
typedef struct _ODLinkInfoResult {  
    ODLinkInfoAction    action;  
    ODBoolean           autoUpdate;  
} ODLinkInfoResult;
```

ODLinkInfoResult Field - action

action ([ODLinkInfoAction](#))

The action taken by the user to dismiss the dialog box.

ODLinkInfoResult Field - autoUpdate

autoUpdate ([ODBoolean](#))

A flag indicating whether the user chose automatic updating.

kODTrue

The user chose automatic updating.

kODFalse

The user did not choose automatic updating.

ODLinkKey

An opaque 32-bit type used to provide thread-safe access to link content.

```
typedef ODULong ODLinkKey;
```

A link key can be created by the methods [ODLink::Lock](#) and [ODLinkSource::Lock](#). The link key must be used in any method that returns or modifies the content storage unit of a link object or link-source object.

ODLinkStatus

An enumeration specifying the link status of a frame.

```
enum ODLinkStatus {  
    kODInLinkDestination;  
    kODInLinkSource;  
    kODNotInLink;  
};
```

ODLinkStatus Field - kODInLinkDestination

kODInLinkDestination

The frame is embedded in the destination of a link; the content of this frame is thus supplied by a link.

ODLinkStatus Field - kODInLinkSource

kODInLinkSource

The frame is embedded in content that is the source of one or more links, but not in content that is the destination of a link.

ODLinkStatus Field - kODNotInLink

kODNotInLink

The frame is not embedded in any linked content, source or destination.

ODMapping

Pointer to a transform matrix.

```
typedef ODMatrix *ODMapping;
```

ODMenuID

A platform-specific identifier for a menu.

```
typedef ODSShort ODMenuID;
```

To prevent menu IDs from clashing, a convention has been established for root parts, parts and shells. Root parts should restrict their root menu IDs to be in the range ODMENUID_ROOTPART_FIRST to ODMENUID_ROOTPART_LAST. Parts should restrict their menu IDs of menu items they create for display to be in the range ODMENUID_ACTIVEPART_FIRST to ODMENUID_ACTIVEPART_FIRST. Shells should restrict their values in the range ODMENUID_SHELL_FIRST to ODMENUID_SHELL_LAST. If this convention is not adhered to, there is a risk that the part and the shell might both create menu items on the same menu using the same menu ID. If this occurs, ambiguity and misdirected menu modifications may result.

OpenDoc has two other predefined meta menu IDs. ODMENUID_ROOT represents the parent menu item of the top-level menu, and ODMENUID_ALL represents the entire menu tree.

ODMenuItem

A menu item.

```
typedef MENUITEM ODMMenuItem;
```

ODMenuItemID

A platform-specific identifier for a menu item.

```
typedef ODSShort ODMMenuItemID;
```

ODName

A name in international text.

```
typedef ODIText ODName;
```

ODNSTypeSpec

An enumeration specifying the types of data that a name space applies to.

```
enum ODNSTypeSpec {  
    ODNSDataTypeODObject;  
    ODNSDataTypeODValue;  
};
```

ODNSTypeSpec Field - ODNSDataTypeODObject

ODNSDataTypeODObject
An object name space.

ODNSTypeSpec Field - ODNSDataTypeODValue

ODNSDataTypeODValue
A value name space.

ODObjectType

A string specifying the type of persistent object represented by a particular storage unit.

```
typedef ODTType ODOBJECTType;
```

kODFrameObject	A frame.
kODNonPersistentFrameObject	A nonpersistent frame.
kODPartObject	A part.

ODOSAID

OSA identifier.

```
typedef ODULong ODOSAID;
```

ODPasteAsMergeSetting

A 32-bit value used to communicate settings to the [ODClipboard::ShowPasteAsDialog](#) and [ODDragAndDrop::ShowPasteAsDialog](#) methods.

```
typedef struct ODPasteAsMergeSetting;
```

Values specify which **At the Destination** radio button (**Merge with Contents** or **Embed As**) is initially selected and whether the other button is available.

kODPasteAsEmbed	Embed As is initially selected; Merge with Contents is available.
kODPasteAsEmbedOnly	Embed As is selected; Merge with Contents is disabled.
kODPasteAsMerge	Merge with Contents is initially selected; Embed As is available.
kODPasteAsMergeOnly	Merge with Contents is selected; Embed As is disabled.

ODPasteAsResult

A structure for storing the results from the Paste As dialog box.

```
typedef struct _ODPasteAsResult {
    ODBoolean    pasteLinkSetting;
    ODBoolean    autoUpdateSetting;
    ODBoolean    mergeSetting;
    ODTypeToken  selectedView;
    ODType       selectedKind;
    ODType       translateKind;
    ODEditor     editor;
} ODPasteAsResult;
```

ODPasteAsResult Field - pasteLinkSetting

pasteLinkSetting (ODBoolean)

A flag indicating whether a link was chosen

kODTrue

A link was chosen.

kODFalse

A link was not chosen.

ODPasteAsResult Field - autoUpdateSetting

autoUpdateSetting (ODBoolean)

A flag indicating whether automatic link updates were chosen. This parameter is relevant only if the *pasteLinkSetting* parameter is kODTrue.

kODTrue

Automatic link updates were chosen.

kODFalse

Automatic link updates were not chosen.

ODPasteAsResult Field - mergeSetting

mergeSetting (ODBoolean)

A flag indicating whether incorporation or embedding was chose.

kODTrue

Incorporation was chosen

kODFalse

Embedding was chosen.

ODPasteAsResult Field - selectedView

selectedView (ODTypeToken)

The view type chosen.

ODPasteAsResult Field - selectedKind

selectedKind (ODType)

The part kind chosen for merging.

ODPasteAsResult Field - translateKind

translateKind ([ODType](#))

If data translation was chosen, this parameter indicates the available type that should be translated to the selected kind. If an available kind was chosen, this field will be kODNULL.

ODPasteAsResult Field - editor

editor ([ODEditor](#))

The preferred editor to bind to the part after embedding or kODNoEditor if no specific editor was chosen. This parameter is relevant only if the *mergeSetting* parameter is kODFalse.

ODPersistentObjectID

A 32-bit ID of a part or frame, used for scripting.

```
typedef ODULong ODPersistentObjectID;
```

ODPlatformCanvas

A 32-bit wrapper for a pointer to a platform-specific or graphics-system-specific drawing environment.

```
typedef unsigned long ODPlatformCanvas;
```

ODPlatformDragReference

Drag reference.

```
typedef ULONG ODPlatformDragReference;
```

ODPlatformMenu

A 32-bit wrapper for a platform-specific structure representing a menu.

```
typedef unsigned long ODPlatformMenu;
```

ODPlatformMenuBar

A 32-bit wrapper for a platform-specific structure representing a menu bar.

```
#define unsigned long ODPlatformMenuBar
```

ODPlatformMenuItem

Platform menu item.

```
typedef struct _ODPlatformMenuItem {  
    short          iPosition;    /* Position. */  
    unsigned short afStyle;      /* Style. */  
    unsigned short afAttribute;  /* Attribute. */  
    unsigned short id;           /* Identity. */  
    unsigned long  hwndSubMenu;  /* Handle to the submenu. */  
    unsigned long  hItem;        /* Item. */  
} ODPlatformMenuItem;
```

ODPlatformMenuItem Field - iPosition

iPosition (short)
Position.

ODPlatformMenuItem Field - afStyle

afStyle (unsigned short)
Style.

ODPlatformMenuItem Field - afAttribute

afAttribute (unsigned short)
Attribute.

ODPlatformMenuItem Field - id

id (unsigned short)
Identity.

ODPlatformMenuItem Field - hwndSubMenu

hwndSubMenu (unsigned long)
Handle to the submenu.

ODPlatformMenuItem Field - hItem

hItem (unsigned long)
Item.

ODPlatformPrintJob

A 32-bit wrapper for a pointer to a platform-specific or graphics-system-specific print job data structure.

```
typedef unsigned long ODPlatformPrintJob;
```

When a part creates a print job, it creates an [ODPlatformCanvas](#) object and initializes it with the presentation space handle for the printer. It then creates an [ODCanvas](#) object and initializes this using the [ODPlatformCanvas](#) object. The part must also create and initialize an [ODPlatformPrintJob](#) structure and call the canvas' [GetPlatformPrintJob](#) method.

ODPlatformShape

A 32-bit wrapper for a pointer to graphics-system-specific shape data.

```
typedef unsigned long ODPlatformShape;
```

The data format is unspecified, and it must be tagged with an [ODGraphicsSystem](#) value to identify the graphics system to which it belongs. Given the pair (ODPlatformShape, [ODGraphicsSystem](#)), it is always possible to identify the exact type of the shape data.

ODPlatformTransform

A 32-bit wrapper for graphics-system-specific transformation data.

```
typedef unsigned long ODPlatformTransform;
```

The data format is unspecified, and it must be tagged with an [ODGraphicsSystem](#) value to identify the graphics system to which it belongs. Given the pair (ODPlatformTransform, [ODGraphicsSystem](#)), it is always possible to identify the exact type of the transformation data.

ODPlatformType

A 32-bit wrapper for the platform-specific type used to identify data formats for data interchange.

```
typedef unsigned long ODPlatformType;
```

ODPlatformTypeSpace

A 32-bit value used to specify the type of a platform-specific structure identifying a type space (data or file).

```
typedef ODULong ODPlatformTypeSpace;
```

kODPlatformDataType
kODPlatformFileType

The native operating system scrap type.
The native operating system file type.

ODPlatformTypeSpaceConsts

An enumeration specifying types of platform-specific structures.

```
enum ODPlatformTypeSpaceConsts {  
    kODPlatformFileType;  
    kODPlatformDataType;  
};
```

ODPlatformTypeSpaceConsts Field - kODPlatformFileType

kODPlatformFileType
The file type of the native operating system.

ODPlatformTypeSpaceConsts Field - kODPlatformDataType

kODPlatformDataType
The data type of the native operating system.

ODPlatformWindow

A 32-bit wrapper for a platform-specific structure representing a window.

```
typedef unsigned long ODPlatformWindow;
```

ODPlatformWindowCreateOptions

Options for creating platform windows.

```
typedef unsigned long ODPlatformWindowCreateOptions;
```

This data type can have one of the following values:

FCF_ACCELTABLE	An accelerator-table resource is required.
FCF_SHELLPOSITION	The size, coordinates, and position is determined by the system.
FCF_STANDARD	A menu template is required.

ODPoint

A structure representing a spatial point in a window or document.

```
typedef struct _ODPoint {  
    ODCoordinate    x; /* X-coordinate. */  
    ODCoordinate    y; /* Y-coordinate. */  
} ODPoint;
```

In two-dimensional imaging models (the only imaging models that currently exist for OpenDoc), a point is represented as a pair of [ODCoordinate](#) values.

ODPoint Field - x

x ([ODCoordinate](#))
X-coordinate.

ODPoint Field - y

y ([ODCoordinate](#))
Y-coordinate.

ODPolygon

A two-dimensional shape composed of one or more contours.

```
typedef ODByteArray ODPolygon;
```

Polygons with multiple contours can be composed of disjoint pieces, or can have interior holes. The *_buffer* field of a polygon byte array points to a 32-bit signed value, indicating the number of contours in the polygon, followed by the contour data for the specified number of contours as described for the [ODContour](#) byte array. Note that the polygon data does not contain [ODContour](#) byte array structures, but the actual contour data for each contour.

ODPolygonData

A structure containing the data for a polygon.

```
typedef struct _ODPolygonData {  
    ODSLong      nContours;  
    ODContour    firstContour;  
} ODPolygonData;
```

ODPolygonData Field - nContours

nContours ([ODSLong](#))
The number of contours.

ODPolygonData Field - firstContour

firstContour ([ODContour](#))
The contours that follow after the first..

ODPositionCode

A 32-bit value used to specify the position of a draft within a document, or the position of a property or value that defined the focus context for a storage unit.

```
typedef ODULong ODPositionCode;
```

Draft Position Codes

The following constants of the `ODPositionCode` type represent positions of one draft in a document relative to another draft of the same document. The drafts of a document can be thought of as a stack with the most recent on top; a given draft is said to be above an earlier draft and below a more recent draft.

<code>kODPosFirstAbove</code>	The draft above (immediately more recent than) the specified draft.
<code>kODPosFirstBelow</code>	The draft below (immediately previous to) the specified draft.
<code>kODPosLastAbove</code>	The draft above (immediately more recent than) the specified draft.
<code>kODPosLastBelow</code>	The draft below (immediately previous to) the specified draft.

kODPosSame	The same draft as the specified draft.
kODPosTop	The top (most recent) draft in the document.

Storage-Unit Position Codes

The following constant of the `ODPositionCode` type represent positions of properties and values within a storage unit; they are used to define or to change the focus context of a storage unit.

kODPosAll	All properties or all values.
kODPosFirstSib	The first property of the storage unit or the first value of the specified property.
kODPosLastSib	The last property of the storage unit or the last value of the specified property.
kODPosMWrap	Wraps iteration of properties or values.
kODPosNextSib	The next property of the storage unit or the next value of the specified property, relative to the current focus context.
kODPosPrevSib	The previous property of the storage unit or the previous value of the specified property, relative to the current focus context.
kODPosSame	The same property or value context as in the current focus context.
kODPosUndefined	Undefined property or value context; typically used when a property name specifies the property and when a value type or value index specifies the value.

Unused Position Codes

The following constants of type the `ODPositionCode` type are reserved for future use.

kODPosReserved11	Reserved for future use.
kODPosReserved12	Reserved for future use.
kODPosReserved13	Reserved for future use.
kODPosReserved14	Reserved for future use.
kODPosReserved15	Reserved for future use.

ODPropertyName

A name for a property within a storage unit.

```
typedef ODISOStr ODPropertyName;
```

Prefixes

The following constants are prefixes used in property names; they specify the type of information stored in the property.

kODPropPreAnnotation	A prefix in the property names of all annotation properties (value "OpenDoc:Annotation:"). An annotation is a property that should be copied automatically whenever the storage unit is cloned. Part-editor developers can use this string as a prefix in the names of annotation properties they define.
kODPropPreODMetaData	A prefix used in the property names of all OpenDoc metadata properties. <i>Metadata</i> is information about the data itself, such as the time it was last modified. Part-editor developers should not use this prefix.

Draft Properties

The following properties save the specified information about the draft, link manager, and window state.

kODPropDocumentName	A user-readable name for a document, of type ODName .
kODPropDraftComment	Any user-created comments on the draft; value type <code>KODOS2IText</code> .

kODPropDraftName	A name for a draft, of type ODISOStr .
kODPropDraftNumber	The number of this draft; value type kODSLong.
kODPropDraftSavedDate	The draft-saved date for drafts that were saved via the Draft History dialog box; value type kODTime_T.
kODPropEditionID	The last edition file ID used by this draft of the document; value type kODULong.
kODPropOriginalID	The original ID of a cloned object; used by the draft cloning methods; value type kODULong.
kODPropRootFrameList	A list of strong references to the root frames of saved windows; value type kODStrongStorageUnitRefs.
kODPropRootPartSU	A strong reference to the storage unit of the root part of this draft; value type kODStrongStorageUnitRef.
kODPropSectionID	The last section ID used by the current draft of the document; value type kODULong.
kODPropStorageUnitName	A name for a storage unit, of type ODISOStr .

Persistent-Object Properties

The following properties are used by persistent objects of all kinds.

kODPropObjectType	The type of persistent object (for example, part, frame, link); value type kODObjectType.
kODPropStorageUnitType	The type of storage unit (for example, the storage unit for a frame object, for link content, and so forth); value type kODISOStr.

Part Properties

The following properties save the specified information about parts.

kODPropComments	Any user-created comments about the part; value type kODOS2IText).
kODPropContents	The stored intrinsic data for the part; the value types correspond to part kinds supported by the part.
kODPropCreateDate	The creation date and time of the object; value type kODTime_T.
kODPropCustomIcon	The custom icon for a part; value type
kODPropIconFamily	The data used for the custom icon of a part on the General page of the Properties notebook.
kODPropDisplayFrames	A list of weak references to the display frames of this part; value type kODWeakStorageUnitRefs.
kODPropHistory	The data used for the history information on the General page of the Properties notebook.
kODPropIsStationery	A Boolean specifying whether this part is a template part; value type kODBoolean.
kODPropKeyPhrases	The data used for the key phrases information on the File page of the Properties notebook.
kODPropModDate	The date and time the part was last modified; value type kODTime_T.
kODPropModUser	The user name when the part was last modified; value type kODOS2IText.
kODPropName	The name of the part; value type kODOS2IText.
kODPropPreferredEditor	The editor ID of the preferred editor (the editor that last wrote this part to persistent storage); value type kODEditor.
kODPropPreferredKind	The value in the kODPropContents property that should be read by an editor bound to this part; value type kODISOStr. This

property, if present, overrides the ordering of values in the kODPropContents property.

Part Containing Properties

The following properties are standardized properties for part containers.

kODBackgroundColor
kODBackgroundTransparency
kODFont
kODForegroundColor
kODTransparencySupported

value type kODRGB2
value type kODBoolean.
The font is of type kODFontNameSize.
value type kODRGB2
value type kODBoolean.

Frame Properties

The following properties save the specified information about frames.

kODPropBiasTransform

The bias transform to be attached to the canvas on which this frame is drawn; value type kODTransform.

kODPropContainingFrame

A weak reference to the containing frame of this frame; value type kODWeakStorageUnitRef.

kODPropDoesPropagateEvent

A boolean specifying whether this frame's part propagates events; value type kODBoolean.

kODPropFrameGroup

The group ID of the frame group to which this frame belongs; value type kODULong.

kODPropFrameShape

The frame shape for this frame; value type kODPolygon.

kODPropGraphicsSystem

The graphics system used to draw in this frame's part; value type kODSShort. The stored data is interpreted as type [ODGraphicsSystem](#).

kODPropInternalTransform

The internal transform for this frame; value type kODTransform.

kODPropIsFrozen

A boolean specifying whether this frame is bundled; value type kODBoolean.

kODPropIsOverlaid

A boolean specifying whether this frame is overlaid; value type kODBoolean.

kODPropIsRoot

A boolean specifying whether this frame is the root frame of a window; value type kODBoolean.

kODPropIsSubframe

A boolean specifying whether this frame is a subframe of another frame; value type kODBoolean.

kODPropLinkStatus

The link status of this frame; value type kODULong, interpreted as type [ODLinkStatus](#).

kODPropPart

A strong reference to the part displayed in this frame; value type value type kODStrongStorageUnitRef.

kODPropPartInfo

The part info (part-specific data) stored with this frame. The value type is determined by the part editor.

kODPropPresentation

The presentation of the part displayed in this frame; value type kODISOSTr.

kODPropSequenceNumber

The sequence number of this frame within its frame group; value type kODULong.

kODPropViewType

The view type of the part displayed in this frame; value type kODISOSTr.

kODPropWindowProperties

A strong reference to a storage unit containing window size and so forth; value type value type kODStrongStorageUnitRef. This property is added to the root frame of saved windows.

Window Properties

The following properties save the specified information about windows.

kODPropRootFrame

A strong reference to the root frame; value type value type

	kODStrongStorageUnitRef.
kODPropShouldShowLinks	A boolean specifying whether parts in this window should display link borders; value type kODBoolean.
kODPropSourceFrame	A strong reference to the source frame; value type kODStrongStorageUnitRef.
kODPropWindow	A weak reference to a window's storage unit; value type kODWeakStorageUnitRef.
kODPropWindowCreateFlags	OS/2 window create flags; value type kODULong.
kODPropWindowHasCloseBox	A boolean specifying whether this window has a close box; value type kODBoolean.
kODPropWindowHasZoomBox	A boolean specifying whether this window has a zoom box; value type kODBoolean.
kODPropWindowIsFloating	A boolean specifying whether this window can float; value type kODBoolean.
kODPropWindowIsResizable	A boolean specifying whether this window has a resize box; value type kODBoolean.
kODPropWindowIsRootWindow	A boolean specifying whether this window is the root window; value type kODBoolean.
kODPropWindowIsVisible	A boolean specifying whether this window is visible; value type kODBoolean.
kODPropWindowRect	The bounding rectangle of a window; value type kODRect.
kODPropWindowRefCon	The window's reference constant, which is set used by your part; value type kODSLong.
kODPropWindowSwpFlags	OS/2 set window position flags; value type SWP .
kODPropWindowTitle	The title of a window; value type kODOS2ITEXT.

Data-Transfer Properties

The following properties save the specified information in the content storage units of data-interchange objects (the clipboard, the drag-and-drop object, link-source objects and link objects).

kODPropContents	The data being transferred, in the same format as in the storage unit of the source part; the value types correspond to the types (part kinds) of the data being transferred.
kODPropContentFrame	A weak reference to the embedded frame being transferred; value type kODWeakStorageUnitRef. This property exists only if the data being transferred consists of a single embedded frame.
kODPropEditionAlias	The alias of an edition file used in implementing cross-document links.
kODPropLinkSection	The section record of a cross-document link.
kODPropLinkSpec	A link specification; value type kODLinkSpec. This property indicates that the destination part may paste a link to the original content being transferred.
kODPropMouseDownOffset	The offset of the location at which a mouse-down event occurred from the top left corner of the selection; value type kODPoint.
kODPropName	If the transferred data is embedded at the destination, the name to be used for the resulting embedded part; value type kODOS2IText.
kODPropOriginalCloneKind	The kind of clone operation that deposited content in this storage unit; value type kODULong, interpreted as type ODCloneKind .
kODPropOriginalDraft	The nonpersistent reference to the original draft for the cloned data in this storage unit; value type kODULong.
kODPropProxyContents	Suggested adornments to apply to the embedded frame being

transferred; value type depends on type of adornment being transferred. This property exists only if the data being transferred consists of a single embedded frame.

kODPropSuggestedFrameShape

If the transferred data is embedded at the destination, the suggested shape for the resulting embedded frame; the value type is either kODPolygon or a platform-specific value type.

kODPropLinkSource

A weak reference to the link-source object associated with this link object; value type kODWeakStorageUnitRef.

kODPropAutoUpdate

A boolean specifying whether this link is to be updated automatically; value type kODBoolean.

kODPropChangeTime

The date and time of this link's last update; value type kODTime_T.

kODPropLink

A weak reference to the link object associated with this link-source object; value type kODWeakStorageUnitRef.

kODPropLinkContentSU

A strong reference to the contents storage for the linked data; value type value type kODStrongStorageUnitRef.

kODPropSourcePart

A weak reference to the part that contains (or last contained) the source data for this link; value type kODWeakStorageUnitRef.

kODPropCloneKindUsed

The kind of cloning operation used to clone objects into this data-transfer object; value type kODCloneKind.

ODPtr

A general-purpose pointer.

```
typedef void *ODPtr;
```

ODPurgePriority

The purge priority.

```
typedef ODUlong ODPurgePriority;
```

kODAllBlocks

All blocks should be purged.

kODInvisibleBlocks

Only blocks with no references to them should be purged.

kODVisibleBlocks

Only blocks with references to them should be purged.

ODPurgePriorityConsts

An enumeration specifying the purge priority.

```
enum ODPurgePriorityConsts {  
    kODAllBlocks;  
    kODInvisibleBlocks;  
    kODVisibleBlocks;  
};
```

```
};
```

ODPurgePriorityConsts Field - kODAllBlocks

kODAllBlocks

All blocks are to be purged.

ODPurgePriorityConsts Field - kODInvisibleBlocks

kODInvisibleBlocks

Only blocks without references to them are to be purged.

ODPurgePriorityConsts Field - kODVisibleBlocks

kODVisibleBlocks

Only blocks with references to them are to be purged.

ODRawPtr

A generic pointer.

```
typedef ODUByte *ODRawPtr;
```

ODRect

A structure representing a rectangle whose sides are aligned with the axes of the current coordinate system.

```
typedef struct _ODRect {  
    ODCoordinate    left;  
    ODCoordinate    top;  
    ODCoordinate    right;  
    ODCoordinate    bottom;  
} ODRect;
```

A rectangle is represented as four [ODCoordinate](#) values. By convention, a rectangle does not include its bottom or right edge; this makes it easier to have adjacent yet nonoverlapping rectangles.

ODRect Field - left

left ([ODCoordinate](#))
The left extent of the rectangle.

ODRect Field - top

top ([ODCoordinate](#))
The top extent of the rectangle.

ODRect Field - right

right ([ODCoordinate](#))
The right extent of the rectangle.

ODRect Field - bottom

bottom ([ODCoordinate](#))
The bottom extent of the rectangle.

ODRespectMarksChoices

An enumeration specifying the possible values for clearing an action history.

```
enum ODRespectMarksChoices {  
    kODDontRespectMarks;  
    kODRespectMarks;  
};
```

ODRespectMarksChoices Field - kODDontRespectMarks

kODDontRespectMarks
Clear the whole action history, ignoring any marks that indicate the beginning of an action subhistory.

ODRespectMarksChoices Field - kODRespectMarks

kODRespectMarks

Clear only the actions within the current subhistory.

ODSByte

A signed, 8-bit value, typically used to represent a single character.

```
typedef char ODSByte;
```

ODSendMode

A context-specific mode associated with an OSA event. A wrapper for the OSA events type AESendMode.

```
typedef ODSLong ODSendMode;
```

ODSendPriority

A priority specifying whether an OSA event is put at the back of the event queue or at the front of the queue. A wrapper for the OSA events type AESendPriority.

```
typedef ODSShort ODSendPriority;
```

ODSiblingOrder

An enumeration used to specify the order in which siblings are processed when iterating through a facet hierarchy. Constant descriptions

```
enum ODSiblingOrder {  
    kODBackToFront;  
    kODFrontToBack;  
};
```

ODSiblingOrder Field - kODBackToFront

kODBackToFront

Process the siblings in a facet hierarchy from back to front.

ODSiblingOrder Field - kODFrontToBack

kODFrontToBack

Process the siblings in a facet hierarchy from front to back.

ODSize

An unsigned 32-bit integer value used to specify the size of a data type, a buffer, or a memory block.

```
typedef ODULong ODSIZE;
```

ODSLong

A signed, 32-bit integer value.

```
typedef long ODSLong;
```

ODSShort

A signed, 16-bit integer value.

```
typedef short ODSShort;
```

ODStorageUnitID

An identifier for a storage unit.

```
typedef ODID ODStorageUnitID;
```

KODNULLID

A null storage-unit ID.

ODStorageUnitKey

A 32-bit value used as an access key used for locking and unlocking a storage unit.

```
typedef ODULong ODStorageUnitKey;
```

kODNULLKey A storage unit key has not yet been granted, or could not be granted.

ODStorageUnitName

A name for a storage unit.

```
typedef ODISOStr ODStorageUnitName;
```

ODStorageUnitRef

An opaque type representing a persistent reference for a storage unit. Whereas a value of the [ODStorageUnitID](#) type identifies a storage unit within the current session, a value of the ODStorageUnitRef type identifies it persistently across sessions.

A value of the [ODStorageUnitID](#) type is created by a storage unit, which must be focused on the value for which it was created; if you store it in a different value, it will almost certainly not refer to the correct storage unit. For more information on persistent references to storage units, see the chapter on storage in *OpenDoc Programming Guide* .

```
typedef ODID ODStorageUnitRef;
```

kODStorageUnitRefSize The size (number of bytes) of an ODStorageUnitRef.

ODTime

A 32-bit value representing a point in time, as the number of seconds elapsed since the beginning of 1970.

```
typedef ODForeign ODTime;
```

ODTime2

A 32-bit time designation.

```
typedef ODTime ODTime2;
```

ODToken

(Obsolete) A 32-bit value used to represent tokens.

```
typedef ODULong ODTOKEN;
```

ODTradITextDataHeader

```
typedef struct _ODTradITextDataHeader {  
    ODScriptCode    theScriptCode;  
    ODLangCode      theLangCode;  
} ODTradITextDataHeader;
```

ODTradITextDataHeader Field - theScriptCode

theScriptCode (ODScriptCode)
The script code; the default is 0.

ODTradITextDataHeader Field - theLangCode

theLangCode (ODLangCode)
The language code; the default is 0.

ODTradITextData

```
typedef struct _ODTradITextData {  
    ODScriptCode    theScriptCode;  
    ODLangCode      theLangCode;  
    char            theText[1];  
} ODTradITextData;
```

ODTradITextData Field - theScriptCode

theScriptCode (ODScriptCode)
The script code; the default is 0.

ODTradlTextData Field - theLangCode

theLangCode ([ODLangCode](#))

The language code; the default is 0.

ODTradlTextData Field - theText[1]

theText[1] (char)

An array of characters.

ODTransformType

A 16-bit value used to specify a kind of transformation specified by a transform.

```
typedef ODSShort ODTransformType;
```

kODIdentityXform	Identity (no-op) transform.
kODInvalidXform	Bad matrix [internal use only].
kODLinearTranslateXform	Scale, rotate, skew, and translation.
kODLinearXform	Scale, rotate, and skew.
kODPerspectiveXform	Perspective transformation, which applies a 3D or distortion effect.
kODScaleTranslateXform	Scale and translation (offset).
kODScaleXform	Pure scale.
kODTranslateXform	Pure translation (offset).
kODUnknownXform	Type not known yet [internal use only].

ODTranslateResult

An enumeration specifying the possible results of a translation.

```
typedef ODULong ODTranslateResult;
```

kODCanTranslate	Translation is allowed with the given types.
kODCannotTranslate	Translation is not allowed with the given types.

ODTranslateResultConsts

An enumeration specifying the possible results of a translation.

```
enum ODTranslateResultConsts {  
    kODCannotTranslate;  
    kODCanTranslate;  
    kODNative;  
};
```

ODTranslateResultConsts Field - kODCannotTranslate

kODCannotTranslate

Translation is not allowed with the specified type.

ODTranslateResultConsts Field - kODCanTranslate

kODCanTranslate

Translation is allowed with the specified type.

ODTranslateResultConsts Field - kODNative

kODNative

The part handles the data directly.

ODTraversalType

An enumeration used to specify the order of iteration through a facet hierarchy. The root of the hierarchy is the facet whose embedded facets are being traversed. If that facet's sibling order is front to back, sibling facets at the same level in the hierarchy are traversed starting with the frontmost; if the sibling order is back to front, sibling facets are traversed starting with the backmost.

```
enum ODTraversalType {  
    kODBottomUp;  
    kODChildrenOnly;  
    kODTopDown;  
};
```

ODTraversalType Field - kODBottomUp

kODBottomUp

Traverse the facet tree from the bottom up, visiting a parent facet after visiting all its children. If sibling order is front to back, traversal starts with the frontmost facet at the lowest level in the hierarchy; if back to front, at the backmost facet at the lowest level.

ODTraversalType Field - kODChildrenOnly

kODChildrenOnly

Traverse only the children of the specified facet.

ODTraversalType Field - kODTopDown

kODTopDown

Traverse the facet tree from the top down.

ODType

A string used generically within OpenDoc to represent drag-image types, object types and value types for storage units, frame presentation types and view types, focus types, and extension types.

```
typedef ODISOStr ODType;
```

The following constants are defined for this datatype:

Frame Presentations

kODPresDefault

The default presentation for a frame.

Frame View Types

kODViewAsFrame

Framed view type.

kODViewAsLargelcon

Large (standard)-icon view type.

kODViewAsSmalllcon

Small-icon view type.

kODViewAsThumbnail

Thumbnail view type.

Extensions

kODExtSemanticInterface

An extension to support a semantic interface in your part.

kODExtStatusLine

An extension to support a status line in your part.

kODMenuExtension

An extension that allows parts to share menu bar space.

kODSettingsExtension

An extension to add panels to the part info dialog box.

ODTypeToken

A 32-bit value used to represent the tokenized form of an [ODType](#) value.

```
typedef ODULong ODTypeToken;
```

kODNullTypeToken	A null type token.
kODNullFocus	No focus. (This value is returned by focus iterators.)

ODUByte

An unsigned 8-bit value.

```
typedef octet ODUByte;
```

ODULong

An unsigned 32-bit value.

```
typedef unsigned long ODULong;
```

ODUpdateID

A 32-bit value used as an update identifier for clipboard content or linked content.

```
typedef ODULong ODUpdateID;
```

Two ODUpdateID values associated with different versions of the same content may be tested for equality, but any other use of these values is meaningless.

kODUnknownUpdate	A value guaranteed to be different from any actual update ID. This constant can be used by parts when the update ID associated with shared content is unknown.
------------------	--

ODUShort

An unsigned 16-bit value.

```
typedef unsigned short ODUShort;
```

ODValue

An opaque type representing a value of a storage unit.

```
typedef void *ODValue;
```

ODValueIndex

A 32-bit integer used as an index for a value within a property of a storage unit. The first value created for a property has index 1; the second, 2; and so on.

```
typedef ODID ODValueIndex;
```

Possible values are shown in the following list:

kODIndexAll	Any value index. This constant is used when focusing a storage unit on a property and when checking for the existence of a property in a storage unit.
-------------	--

ODValueType

A string used to identify the type of data in the value of a storage unit.

```
typedef ODType ODValueType;
```

Possible values are shown in the following list:

kODBoolean	Type ODBoolean .
kODCloneKind	Type ODCloneKind .
kODDate	Type CDATE .
kODDragItem	A PM DRAGITEM structure.
kODDragOperation	A PM drag operation.
kODEditor	Type ODEditor .
kODFileType	The name of the dragged file.
kODFileTypeEA	The extended attributes of the dragged file.
kODFontNameSize	A string containing the font size and name in the same format as the OS/2 PRESPARAMS .
kODIconFamily	Type ODIconFamily .
kODIconFamilyWin	Type ODIconFamily representing a Windows icon family.
kODIconFamilyOS2	Type ODIconFamily representing an OS/2 icon family.
kODIconFamilyAIX	Type ODIconFamily , representing an AIX icon family.
kODIntlText	Type ODIText , converted to a document-interchange format that corresponds to ISO standard 10646-1, 1993, fully decomposed.
kODISOStr	Type ODISOStr .
kODISOStrList	A list of ISO strings.
kODLinkSpec	An ODLinkSpec object.
kODOBJECTType	Type ODOBJECTType .
kODPoint	Type ODPoint .
kODOS2IText	Type ODIText .
kODPolygon	Type ODPolygon .

kODRect	Type ODRect .
kODRGB2	Type RGB.
kODSelectedKind	The selected kind for the current drag-and-drop operation.
kODSelectedRMF	The selected RMF pair for the current drag-and-drop operation.
kODSLong	Type ODSLong .
kODSShort	Type ODSShort .
kODStrongStorageUnitRef	Type ODStorageUnitRef representing a strong storage-unit reference.
kODStrongStorageUnitRefs	A list of strong storage-unit references.
kODTime_T	Type ODTime .
kODTransform	Type ODMatrix
kODTypeAll	All value types. This constant is used when focusing a storage unit on a property and when checking for the existence of a property in a storage unit. The constant kODTypeAll is equivalent to kODNULL. Their different names provide clarity to the code that uses them. Typically kODNULL is used to mean an absent or invalid value type; kODTypeAll is used to specify all values of a given property.
kODULong	Type ODULong .
kODUShort	Type ODUShort .
kODWeakStorageUnitRef	Type ODStorageUnitRef representing a weak storage unit reference.
kODWeakStorageUnitRefs	A list of weak storage unit references.

ODVersionID

A 32-bit value indicating the version ID.

```
typedef ODULong ODVersionID;
```

ODWide

A 64-bit signed integer.

```
typedef struct _ODWide {
    ODSLong    hi;
    ODULong    lo;
} ODWide;
```

ODWide Field - hi

hi (ODSLong)
The high-order 32-bit signed integer.

ODWide Field - lo

lo (ODULong)
The low-order 32-bit unsigned integer.

OperatingSystem

Operating-system enumeration.

```
typedef enum OperatingSystem;
```

OSAEvent

List of attributes and parameters necessary for an OSA event.

```
typedef AERecord OSAEvent;
```

PAGEINFO

Settings page information structure.

```
typedef struct _PAGEINFO {  
    ULONG        cb;                /* Length of PAGEINFO structure. */  
    HWND         hwndPage;          /* Handle of page. */  
    PFNWP        pfnwp;             /* Window procedure. */  
    ULONG        resid;             /* Resource identity. */  
    PVOID        pCreateParams;     /* Pointer to creation parameters. */  
    USHORT       dlgid;             /* Dialog identity. */  
    USHORT       usPageStyleFlags;  /* Notebook control-page style flags. */  
    USHORT       usPageInsertFlags; /* Notebook control-page insertion flags. */  
    USHORT       usSettingsFlags;   /* Settings flag. */  
    PSZ          pszName;           /* Pointer to a string containing the page name. */  
    USHORT       idDefaultHelpPanel; /* Identity of the default help panel. */  
    USHORT       usReserved2;        /* Reserved value; must be zero. */  
    PSZ          pszHelpLibraryName; /* Pointer to the name of the help file. */  
    PUSHORT      pHelpSubtable;     /* Pointer to the help subtable. */  
    HMODULE       hmodHelpSubtable; /* Module handle for the help subtable. */  
    ULONG        ulPageInsertId;    /* Notebook control-page identity. */  
} PAGEINFO;  
  
typedef PAGEINFO *PPAGEINFO;
```

PAGEINFO Field - cb

cb ([ULONG](#))
Length of PAGEINFO structure.

PAGEINFO Field - hwndPage

hwndPage ([HWND](#))
Handle of page.

PAGEINFO Field - pfnwp

pfnwp ([PFNWP](#))
Window procedure.

PAGEINFO Field - resid

resid ([ULONG](#))
Resource identity.

PAGEINFO Field - pCreateParams

pCreateParams ([PVOID](#))
Pointer to creation parameters.

PAGEINFO Field - dlgid

dlgid ([USHORT](#))
Dialog identity.

PAGEINFO Field - usPageStyleFlags

usPageStyleFlags (USHORT)
Notebook control-page style flags.

PAGEINFO Field - usPageInsertFlags

usPageInsertFlags (USHORT)
Notebook control-page insertion flags.

PAGEINFO Field - usSettingsFlags

usSettingsFlags (USHORT)
Settings flag.

This flag must be set to one of the following values:

0
You will not get page numbers.

SETTINGS_PAGE_NUMBERS
Page numbers will automatically be put on the status line for pages that have minor pages under the major tab page.

If you want to use the page numbers, make sure all pages have this setting.

PAGEINFO Field - pszName

pszName (PSZ)
Pointer to a string containing the page name.

PAGEINFO Field - idDefaultHelpPanel

idDefaultHelpPanel (USHORT)
Identity of the default help panel.

PAGEINFO Field - usReserved2

usReserved2 (USHORT)
Reserved value; must be zero.

PAGEINFO Field - pszHelpLibraryName

pszHelpLibraryName ([PSZ](#))

Pointer to the name of the help file.

PAGEINFO Field - pHelpSubtable

pHelpSubtable ([PUSHORT](#))

Pointer to the help subtable.

PAGEINFO Field - hmodHelpSubtable

hmodHelpSubtable ([HMODULE](#))

Module handle for the help subtable.

PAGEINFO Field - ulPageInsertId

ulPageInsertId ([ULONG](#))

Notebook control-page identity.

PartHandlerQueryInfo

A structure containing part handler information.

```
typedef struct _PartHandlerQueryInfo {  
    long          cBytes;  
    ISOString     partHandlerName;  
    char          *partHandlerDisplayName;  
    char          *partHandlerClassName;  
    char          *partKindList;  
    char          *ole2ClassId;  
    char          *windowsIconFileName;  
    char          *dllName;  
};
```

PartHandlerQueryInfo Field - cBytes

cBytes (long)
The size of the data structure.

PartHandlerQueryInfo Field - partHandlerName

partHandlerName (ISOString)
The name of the part handler.

PartHandlerQueryInfo Field - partHandlerDisplayName

partHandlerDisplayName (char *)
The name of the display for the part handler.

PartHandlerQueryInfo Field - partHandlerClassName

partHandlerClassName (char *)
The SOM class name for the part handler.

PartHandlerQueryInfo Field - partKindList

partKindList (char *)
A comma-delimited string containing the part kinds supported by the part handler.

PartHandlerQueryInfo Field - ole2ClassId

ole2ClassId (char *)
The OLE2 class ID for this part handler.

PartHandlerQueryInfo Field - windowsIconFileName

windowsIconFileName (char *)
The name of the Windows's icon file used to represent the part handler.

PartHandlerQueryInfo Field - dllName

dllName (char *)
The name of the DLL for the part handler.

PartKindInfo

Part-kind information.

```
typedef struct _PartKindInfo {  
    ISOString    partKindName;  
    string       partKindDisplayName;  
    string       fileNameFilters;  
    string       fileNameTypes;  
    string       categories;  
    string       categoryDisplayName;  
    string       objectID;  
} PartKindInfo;  
  
typedef PartKindInfo *PPartKindInfo;
```

PartKindInfo Field - partKindName

partKindName (ISOString)
A pointer to a string containing the part kind name.

PartKindInfo Field - partKindDisplayName

partKindDisplayName (string)
A string containing the display name of the part kind.

PartKindInfo Field - fileNameFilters

fileNameFilters (string)
A string containing the file extensions that the specified part kind can use; for example, "*.TXT, *.BMP".

PartKindInfo Field - fileNameTypes

fileNameTypes ([string](#))

A string containing the file types that this part kind can manipulate.

PartKindInfo Field - categories

categories ([string](#))

A string containing the categories for this part kind.

PartKindInfo Field - categoryDisplayName

categoryDisplayName ([string](#))

A string containing the user visible category display name.

PartKindInfo Field - objectID

objectID ([string](#))

A string representing the OpenDoc template object ID.

PartKindQueryInfo

A structure containing part-kind information.

```
typedef struct _PartKindQueryInfo {
    long          cBytes;
    ISOString     partKindName;
    char          *partKindDisplayName;
    char          *filenameFilters;
    char          *filenameTypes;
    char          *categories;
} PartKindQueryInfo;
```

PartKindQueryInfo Field - cBytes

cBytes (long)

The size of the data structure.

PartKindQueryInfo Field - partKindName

partKindName ([ISOString](#))

A string containing the part kind name.

PartKindQueryInfo Field - partKindDisplayName

partKindDisplayName (char *)

A string containing the user-visible display name.

PartKindQueryInfo Field - filenameFilters

filenameFilters (char *)

A comma-delimited string containing the file extensions which this part kind supports.

PartKindQueryInfo Field - filenameTypes

filenameTypes (char *)

A comma-delimited string containing the file types which this part kind supports.

PartKindQueryInfo Field - categories

categories (char *)

A comma-delimited string containing the categories which this part kind supports.

PCSZ

Pointer to a constant null-terminated string.

```
typedef const char *PCSZ;
```

PFNWP

Pointer to a window procedure.

This is the standard function definition for window procedures.

```
typedef FNP *PFNP;
```

The first argument (**HWND**) is the handle of the window receiving the message. The second argument (**ULONG**) is a message identifier. The third argument (**MPARAM**) is the first message parameter (mp1). The fourth argument (**MPARAM**) is the second message parameter (mp2). The function returns an **MRESULT**. Each message has a specific set of possible return codes. The window procedure must return a value that is appropriate for the message being processed.

In the header file, this is a two-part definition as shown below:

```
typedef HRESULT (EXPENTRY FNP) (HWND, ULONG, MPARAM, MPARAM);  
typedef FNP *PFNP;
```

Window procedures must be EXPORTED in the definitions file used by the linker.

PID

Process identity.

```
typedef LHANDLE PID;
```

POINTL

Point structure (long integers).

```
typedef struct _POINTL {  
    LONG    x; /* X-coordinate. */  
    LONG    y; /* Y-coordinate. */  
} POINTL;  
  
typedef POINTL *PPOINTL;
```

POINTL Field - x

x (**LONG**)
X-coordinate.

POINTL Field - y

y (**LONG**)
Y-coordinate.

PRESPARAMS

Presentation parameter data.

```
typedef struct _PRESPARAMS {
    ULONG      cb;          /* Length of the aparam parameter, in bytes. */
    PARAM      aparam[1];  /* Array of presentation attribute parameters. */
} PRESPARAMS;

typedef PRESPARAMS *PPRESPARAMS;
```

PRESPARAMS Field - cb

cb (ULONG)
Length of the *aparam* parameter, in bytes.

PRESPARAMS Field - aparam[1]

aparam[1] (PARAM)
Array of presentation attribute parameters.

PRINTDEST

PRINTDEST data structure.

Contains all the parameters required to issue DevPostDeviceModes and DevOpenDC function calls.

```
typedef struct _PRINTDEST {
    ULONG      cb;          /* Length of data structure, in bytes. */
    LONG       lType;       /* Type of device context. */
    PSZ        pszToken;    /* Device-information token. */
    LONG       lCount;      /* Number of items. */
    PDEVOPENDATA pdopData;  /* Open device context data area. */
    ULONG      fl;          /* Flags. */
    PSZ        pszPrinter;  /* Name of the printer. */
} PRINTDEST;

typedef PRINTDEST *PPRINTDEST;
```

PRINTDEST Field - cb

cb (ULONG)

Length of data structure, in bytes.

PRINTDEST Field - IType

IType ([LONG](#))

Type of device context.

OD_QUEUED

The device context is queued.

OD_DIRECT

The device context is direct.

PRINTDEST Field - pszToken

pszToken ([PSZ](#))

Device-information token.

This is always "".

PRINTDEST Field - ICount

ICount ([LONG](#))

Number of items.

This is the number of items present in the *pdopData* field.

PRINTDEST Field - pdopData

pdopData (PDEVOPENDATA)

Open device context data area.

See [DEVOPENSTRUC](#) for information on the format of *pdopData* .

PRINTDEST Field - fl

fl ([ULONG](#))

Flags.

PD_JOB_PROPERTY

This flag indicates that DevPostDeviceModes should be called with DPDM_POSTJOBPROP before calling

PRINTDEST Field - pszPrinter

pszPrinter (PSZ)

Name of the printer.

A name that specifies the device; for example, "PRINTER1". The name is used for calling DevPostDeviceModes.

The printer device name can be found by calling SplQueryQueue and passing to it the information found in the *pszLogAddress* field of the **DEVOPENSTRUC** structure pointed to by *pdopData*. SplQueryQueue returns a **PRQINFO3** structure. The *pszPrinters* field in **PRQINFO3** contains the printer device name to be used.

PRQINFO3

Print-queue information structure.

This structure is used at information levels 3 and 4.

```
typedef struct _PRQINFO3 {
    PSZ      pszName;           /* Queue name. */
    USHORT   uPriority;         /* Queue priority. */
    USHORT   uStartTime;       /* Minutes after midnight when queue becomes active. */
    USHORT   uUntilTime;       /* Minutes after midnight. when queue ceases to be active. */
    USHORT   fsType;           /* Queue type. */
    PSZ      pszSepFile;        /* Separator-page file. */
    PSZ      pszPrProc;         /* Default queue-processor. */
    PSZ      pszParms;          /* Queue parameters. */
    PSZ      pszComment;        /* Queue description. */
    USHORT   fsStatus;          /* Queue status. */
    USHORT   cJobs;             /* Number of jobs in queue. */
    PSZ      pszPrinters;       /* Print devices connected to queue. */
    PSZ      pszDriverName;     /* Default device driver. */
    PDRIVDATA pDriverData;      /* Default queue job properties. */
} PRQINFO3;

typedef PRQINFO3 *PPRQINFO3;
```

PRQINFO3 Field - pszName

pszName (PSZ)

Queue name.

The maximum length of the name in the network case is 256 (including one byte for zero termination).

PRQINFO3 Field - uPriority

uPriority (USHORT)

Queue priority.

The range is 1 through 9, with 1 being the highest queue priority.

The default job priority (DefJobPrio) is determined from:
 $\text{DefJobPrio} = 100 - (10 * uPriority)$.

If a job is added with **PRJ_NO_PRIORITY** specified, DefJobPrio is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

PRQ_DEF_PRIORITY
 Default priority
PRQ_MAX_PRIORITY
 Highest priority
PRQ_MIN_PRIORITY
 Minimum priority
PRQ_NO_PRIORITY
 No priority.

PRQINFO3 Field - uStartTime

uStartTime (**USHORT**)

Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If *uStartTime* and *uUntilTime* are both 0, the print queue is always available.

PRQINFO3 Field - uUntilTime

uUntilTime (**USHORT**)

Minutes after midnight. when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If *uUntilTime* and *uStartTime* are both 0, the print queue is always available.

PRQINFO3 Field - fsType

fsType (**USHORT**)

Queue type.

PRQ3_TYPE_RAW
 Data is always enqueued in the device specific format.
PRQ3_TYPE_BYPASS
 Allows the spooler to bypass the queue processor and send data directly to the Printer Driver. Setting this bit allows the spooler to print jobs of type PM_Q_RAW while they are still being spooled.
PRQ3_TYPE_APPDEFAULT
 This bit is set for the application default queue only.

PRQINFO3 Field - pszSepFile

pszSepFile (PSZ)

Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

PRQINFO3 Field - pszPrProc

pszPrProc (PSZ)

Default queue-processor.

PRQINFO3 Field - pszParms

pszParms (PSZ)

Queue parameters.

This can be any text string or a NULL string.

PRQINFO3 Field - pszComment

pszComment (PSZ)

Queue description.

A NULL string results in no comment. The maximum length is 48 characters (including one byte for the null terminator).

PRQINFO3 Field - fsStatus

fsStatus (USHORT)

Queue status.

PRQ3_PAUSED

Queue is paused (held).

PRQ3_PENDING

Queue is pending deletion.

PRQINFO3 Field - cJobs

cJobs ([USHORT](#))
Number of jobs in queue.

PRQINFO3 Field - pszPrinters

pszPrinters ([PSZ](#))
Print devices connected to queue.

This cannot be NULL.

PRQINFO3 Field - pszDriverName

pszDriverName ([PSZ](#))
Default device driver.

PRQINFO3 Field - pDriverData

pDriverData ([PDRIVDATA](#))
Default queue job properties.

Note: An application can use *pszDriverName* , *pDriverData* , *pszPrProc* , and *pszParms* to construct a valid DevOpenDC call based only on the queue name.

RGB2

RGB color value.

```
typedef struct _RGB2 {  
    BYTE    bBlue;        /* Blue component of the color definition. */  
    BYTE    bGreen;       /* Green component of the color definition. */  
    BYTE    bRed;         /* Red component of the color definition. */  
    BYTE    fcOptions;    /* Entry options. */  
} RGB2;  
  
typedef RGB2 *PRGB2;
```

RGB2 Field - bBlue

bBlue (BYTE)
Blue component of the color definition.

RGB2 Field - bGreen

bGreen (BYTE)
Green component of the color definition.

RGB2 Field - bRed

bRed (BYTE)
Red component of the color definition.

RGB2 Field - fcOptions

fcOptions (BYTE)
Entry options.

These can be ORed together if required:

PC_RESERVED

The color entry is reserved for animating color with the palette manager.

PC_EXPLICIT

The low-order word of the color table entry designates a physical palette slot. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

PSZ

Pointer to a null-terminated string.

If you are using C++ **, you may need to use PCSZ.

```
typedef unsigned char *PSZ;
```

SemHandle

A generic pointer to a semaphore handle.

```
typedef void *SemHandle;
```

SHORT

Signed integer in the range -32 768 through 32 767.

```
#define SHORT short
```

SIZEL

Size structure (LONG values).

```
typedef struct _SIZEL {  
    LONG    cx; /* Width. */  
    LONG    cy; /* Height. */  
} SIZEL;  
  
typedef SIZEL *PSIZEL;
```

SIZEL Field - cx

cx (LONG)
Width.

SIZEL Field - cy

cy (LONG)
Height.

somToken

A generic SOM type used in IDL files for the user type.

```
typedef void *somToken;
```

Str255

String containing up to 255 characters.

```
typedef somToken Str255;
```

string

A pointer to a char.

```
typedef char *string;
```

SWP

Set-window-position structure.

```
typedef struct _SWP {
    ULONG    fl;           /* Options. */
    LONG     cy;           /* Window height. */
    LONG     cx;           /* Window width. */
    LONG     y;            /* Y-coordinate of origin. */
    LONG     x;            /* X-coordinate of origin. */
    HWND     hwndInsertBehind; /* Window behind which this window is placed. */
    HWND     hwnd;         /* Window handle. */
    ULONG     ulReserved1;  /* Reserved value; must be 0. */
    ULONG     ulReserved2;  /* Reserved value; must be 0. */
} SWP;

typedef SWP *PSWP;
```

SWP Field - fl

fl (ULONG)
Options.

Possible values are shown in the following list:

SWP_ACTIVATE
SWP_DEACTIVATE
SWP_HIDE
SWP_MAXIMIZE
SWP_MINIMIZE
SWP_MOVE
SWP_NOADJUST
SWP_NOERASEWINDOW
SWP_NOREDRAW

SWP_RESTORE

SWP_SHOW

SWP_SIZE

SWP_ZORDER

SWP Field - cy

cy ([LONG](#))
Window height.

SWP Field - cx

cx ([LONG](#))
Window width.

SWP Field - y

y ([LONG](#))
Y-coordinate of origin.

SWP Field - x

x ([LONG](#))
X-coordinate of origin.

SWP Field - hwndInsertBehind

hwndInsertBehind ([HWND](#))
Window behind which this window is placed.

SWP Field - hwnd

hwnd ([HWND](#))
Window handle.

SWP Field - ulReserved1

ulReserved1 ([ULONG](#))
Reserved value; must be 0.

SWP Field - ulReserved2

ulReserved2 ([ULONG](#))
Reserved value; must be 0.

TValue

A pointer to a stream of bytes.

```
typedef ODUByte *TValue;
```

TypePS

A constant representing the presentation space (PS) type.

```
enum TypePS {  
    NormalPS;  
    DragPS;  
    LockedWindowUpdatePS;  
};
```

TypePS Field - NormalPS

NormalPS
A normal PS.

TypePS Field - DragPS

DragPS

A PS used to provide target feedback to the user during a drag operation.

TypePS Field - LockedWindowUpdatePS

LockedWindowUpdatePS

A presentation space which permits drawing if the WinLockWindowUpdate function has been called.

ULONG

32-bit unsigned integer in the range 0 through 4 294 967 295.

```
typedef unsigned long ULONG;
```

USHORT

Unsigned integer in the range 0 through 65 535.

```
typedef unsigned short USHORT;
```

VOID

A data area of undefined format.

```
#define VOID void
```

WindowProperties

A structure representing window properties.

```
typedef struct _WindowProperties {  
    RECTL      boundsRect;  
    Str255     title;  
    ODULong     createFlags;  
    ODULong     swpFlags;
```

```
ODBoolean    wasVisible;
ODBoolean    isResizable;
ODBoolean    isFloating;
ODBoolean    isRootWindow;
ODBoolean    shouldShowLinks;
ODFrame      *sourceFrame;
} WindowProperties;
```

WindowProperties Field - boundsRect

boundsRect (RECTL)
kODPropWindowRect

WindowProperties Field - title

title (Str255)
kODPropWindowTitle

WindowProperties Field - createFlags

createFlags (ODULong)
kODPropWindowCreateFlags

WindowProperties Field - swpFlags

swpFlags (ODULong)
kODPropWindowSwpFlags

WindowProperties Field - wasVisible

wasVisible (ODBoolean)
kODPropIsVisible

kODTrue

The window is visible.

kODFalse

The window is not visible.

WindowProperties Field - isResizable

isResizable (ODBoolean)

kODPropWindowsResizable

kODTrue

The window is resizable.

kODFalse

The window is not resizable.

WindowProperties Field - isFloating

isFloating (ODBoolean)

kODPropWindowsFloating

kODTrue

The window is floating.

kODFalse

The window is not floating.

WindowProperties Field - isRootWindow

isRootWindow (ODBoolean)

kODPropWindowsRootWindow

kODTrue

The window is the root window.

kODFalse

The window is not the root window.

WindowProperties Field - shouldShowLinks

shouldShowLinks (ODBoolean)

kODPropShouldShowLinks

kODTrue

Links should be highlighted in this window.

kODFalse

Links should not be highlighted in this window.

WindowProperties Field - sourceFrame

Return Codes by Number

This section lists OpenDoc errors in order of their error number and provides an explanation for each error.

0
1
3
4
5
6
7
8
9
15
16
18
20
22
23
24
25
26
28
30
32
33
34
35
38
39
40
41
42
44
46
47
48
51
52
53
54
56
57
59
61
62
65
68
69
70
73
75
76
77
84
85
87
88
89
90
92
95
96
97
98
99
100

101
102
104
105
106
107
108
109
110
111
112
113
114
115
117
118
119
120
121
122
123
126
128
129
131
132
133
135
136
137
138
139
140
141
142
143
144
145
146
147
148
157
160
161
162
163
164
165
166
167
168
169
171
172
173
174
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196

197
198
199
202
223
230
231
232
233
234
235
241
242
243
244
245
246
247
248
249
250
251
253
254
255
256
257
257

Return Codes by Name

This section lists OSA errors in order of their error name and provides an explanation for each error.

kODErrAlreadyImportedLink
kODErrBrokenLink
kODErrBrokenLinkSource
kODErrCannotAcquireFrame
kODErrCannotAcquireLink
kODErrCannotAcquirePart
kODErrCannotAddAction
kODErrCannotAddProperty
kODErrCannotAddType
kODErrCannotAllocateDragItem
kODErrCannotChangePermissions
kODErrCannotCollapseDrafts
kODErrCannotCreateContainer
kODErrCannotCreateFrame
kODErrCannotCreateLink
kODErrCannotCreatePart
kODErrCannotCreateWindow
kODErrCannotEmbed
kODErrCannotFindLinkSource
kODErrCannotFindLinkSourceEdition
kODErrCannotGetExternalLink
kODErrCannotMarkAction
kODErrCannotOpenContainer
kODErrCannotRevealLink
kODErrCloningInProgress
kODErrContainerExists
kODErrContainerNotExists
kODErrCorruptLink
kODErrCorruptLinkSource
kODErrCorruptLinkSpecValue
kODErrDispatcherNotInitialized
kODErrDocAlreadyOpened
kODErrDocNotRegistered
kODErrDocNotOpened
kODErrDocNotSaved
kODErrDoesNotDrop
kODErrDoesNotLink

kODErrDoesNotUndo
kODErrDraftNotExists
kODErrDragItemNotFound
kODErrDragTrackingException
kODErrFocusAlreadyRegistered
kODErrFocusIsNonExclusive
kODErrFocusNotRegistered
kODErrEmptyStack
kODError
kODErrExistingDispatchModule
kODErrIllegalClipboardCloneKind
kODErrIllegalNullDispatchModuleInput
kODErrIllegalNullFacetInput
kODErrIllegalNullFrameInput
kODErrIllegalNullInput
kODErrIllegalNullPartInput
kODErrIllegalNullPropertyInput
kODErrIllegalNullShapeInput
kODErrIllegalNullStorageSystemInput
kODErrIllegalNullStorageUnitInput
kODErrIllegalNullSUCursorInput
kODErrIllegalNullTokenInput
kODErrIllegalNullTransformInput
kODErrIllegalRecursiveEmbedding
kODErrInconsistentCloneKind
kODErrInvalidAuxState
kODErrInvalidAviSvr
kODErrInvalidBelowDraft
kODErrInvalidCanvas
kODErrInvalidCloneKind
kODErrInvalidContainer
kODErrInvalidDestinationDraft
kODErrInvalidDocID
kODErrInvalidDocPathName
kODErrInvalidDocument
kODErrInvalidDraft
kODErrInvalidDraftID
kODErrInvalidDraftKey
kODErrInvalidExtension
kODErrInvalidFacet
kODErrInvalidFrame
kODErrInvalidGraphicsSystem
kODErrInvalidID
kODErrInvalidLink
kODErrInvalidLinkID
kODErrInvalidLinkKey
kODErrInvalidLinkSource
kODErrInvalidName
kODErrInvalidObjectType
kODErrInvalidParameter
kODErrInvalidPermissions
kODErrInvalidPersistentObject
kODErrInvalidPersistentObjectID
kODErrInvalidProperty
kODErrInvalidRefCount
kODErrInvalidShell
kODErrInvalidStorageUnit
kODErrInvalidStorageUnitKey
kODErrInvalidStorageUnitRef
kODErrInvalidTargetID
kODErrInvalidType
kODErrInvalidValue
kODErrIteratorNotInitialized
kODErrIteratorOutOfSync
kODErrKeyAlreadyExists
kODErrLinkAlreadyExported
kODErrLinkAlreadyOpened
kODErrLinkIDNotInDoc
kODErrLinkNotOpened
kODErrLinkNotRegistered
kODErrMoveIntoSelf
kODErrNoBeginAction
kODErrNoDocumentProperties
kODErrNoDraftProperties
kODErrNoDragManager
kODErrNoDragSystemStorage

kODErrNoEditionManager
kODErrNoLinkContent
kODErrNoLinkSpecValue
kODErrNonEmptyDraft
kODErrNoPreviousDraft
kODErrNoPromises
kODErrNoShapeGeometry
kODErrNotExportedLink
kODErrNotImplemented
kODErrNotImportedLink
kODErrNotRootFrame
kODErrNoValueAtThatIndex
kODErrNoWindow
kODErrNullDestinationFrame
kODErrNullFacetInput
kODErrNullLinkInfoInput
kODErrNullLinkInfoResultInput
kODErrNullPasteAsResultInput
kODErrObjectNotInitialized
kODErrOutOfMemory
kODErrOutstandingDraft
kODErrPartInUse
kODErrPError
kODErrReadErr
kODErrRefCountNotEqualOne
kODErrReleaseMutexInvalidHandle
kODErrReleaseMutexNotOwner
kODErrRequestMutexInterrupt
kODErrRequestMutexInvalidHandle
kODErrRequestMutexOwnerDied
kODErrRequestMutexTimeout
kODErrRequestMutexTooManyRequests
kODErrStorageUnitLocked
kODErrStorageUnitNotLocked
kODErrSubClassResponsibility
kODErrTransformErr
kODErrUndefined
kODErrUnfocusedStorageUnit
kODErrUnknownDragImageType
kODErrUnknownExtension
kODErrUnknownLinkSpecVersion
kODErrUnknownUpdateID
kODErrUnsupportedExtension
kODErrUnsupportedFramePositionCode
kODErrUnsupportedPosCode
kODErrValueIndexOutOfRange
kODErrWaitingUnlock
kODErrWriteErr
kODErrZeroRefCount
kODMaxError
kODMaxLinkError
kODNoError
kODPartNotWrapper

0

0

kODNoError

No error occurred.

1

1 **kODerrUndefined**
An undefined error occurred.

3

3 **kODerrInvalidContainer**
The specified draft is not valid.

4

4 **kODerrInvalidDocument**
The specified document is not valid.

5

5 **kODerrInvalidDraft**
The specified draft is not valid.

6

6 **kODerrInvalidStorageUnit**
The specified storage unit is not valid or the specified persistent object has no storage unit.

7

7 **kODerrCannotEmbed**
Attempt to access embedded frames for a part that does not support embedding.

8

8 **kODerrIllegalNullPartInput**

A frame method was passed null for the part parameter that can not be null.

9

9 **kODerrInvalidFrame**

The specified frame is not valid in the context in which it is used. For example, it is not a display frame of the part whose method is being called.

15

15 **kODerrIllegalNullDispatchModuleInput**

A dispatcher method was passed null for the dispatcher module.

16

16 **kODerrInvalidType**

A value type parameter is unknown or null.

18

18 **kODerrInvalidName**

A name-space name is unknown or null.

20

20 **kODerrInvalidID**

The specified draft identifier is not valid in the context in which it is used.

22

22 **kODErrUnsupportedExtension**

Failure to get an object's extension because the object does not support the specified extension.

23

23 **kODErrInvalidValue**

A value parameter is unknown or null.

24

24 **kODErrInvalidProperty**

A property parameter is unknown or null.

25

25 **kODErrUnsupportedPosCode**

The specified position code is not supported in the context in which it is used.

26

26 **kODErrNoValueAtThatIndex**

Failure to focus a storage unit because there is no value at the specified index.

28

28 **kODErrKeyAlreadyExists**

Failure to create a name-space object because a name space with the specified name already exists.

30

30 **kODErrPartInUse**

The part for a specified wrapper is currently in use.

32

32 **kODErrCannotAddProperty**

Unable to add the given property to a storage unit.

33

33 **kODErrCannotAddType**

The specified type could not be added to the storage unit.

34

34 **kODErrUnfocusedStorageUnit**

Attempt to perform an operation on a storage unit that is not properly focused.

35

35 **kODErrInvalidPermissions**

The attempted action is not consistent with existing draft permissions.

38

38 **kODerrInvalidBelowDraft**

Attempt to create a draft below an invalid draft (that is, a draft that is not the top draft of the document).

39

39 **kODerrDraftNotExists**

Failure to get a draft object because no draft exists with the specified identifier.

40

40 **kODerrContainerExists**

Failure to create a container object because a container with the specified identifier already exists.

41

41 **kODerrCannotCollapseDrafts**

Attempt to collapse drafts specified by an invalid range of drafts.

42

42 **kODerrNonEmptyDraft**

Attempt to collapse drafts of a document that are not empty.

44

44 **kODerrContainerNotExists**

Failure to get a container object because no container exists with the specified identifier.

46

46 **kODErrNoPreviousDraft**
There is no previous draft for this document.

47

47 **kODErrIllegalNullPropertyInput**
A storage-unit method was passed null for a property parameter that can not be null.

48

48 **kODErrIllegalNullSUCursorInput**
A storage-unit method was passed null for a storage-unit cursor parameter that can not be null.

51

51 **kODErrNoDraftProperties**
Failure to create the storage unit to store draft properties.

52

52 **kODErrInvalidRefCount**
The persistent-object reference count is not correct.

53

53 **kODErrCannotCreateFrame**
Failure to create the requested frame.

54

54 **kODErrCannotCreateWindow**

The window-state object cannot create a window.

56

56 **kODErrOutstandingDraft**

The attempted action would invalidate an outstanding draft (that is, one that is currently being referenced by some object).

57

57 **kODErrZeroRefCount**

Attempt to decrement an object's reference count that is already 0.

59

59 **kODErrInvalidGraphicsSystem**

This implementation of OpenDoc does not support the specified graphics system, or there is no drawing structure or print job associated with that graphics system.

61

61 **kODErrOutOfMemory**

Not enough memory to perform the specified operation (which involves an allocation).

62

62 **kODErrInvalidDraftID**

The specified draft ID is not valid in the context in which it is used.

65

65 **kODErrCannotChangePermissions**

Attempt to change permissions of a draft that has already been retrieved with different permissions.

68

68 **kODErrCannotAcquireFrame**

Failure to recreate the frame object from the specified storage unit.

69

69 **kODErrCannotCreatePart**

Failure to create the requested part object.

70

70 **kODErrCannotAcquirePart**

Failure to recreate the part object from the specified storage unit.

73

73 **kODErrInvalidStorageUnitRef**

The specified persistent storage-unit reference is not valid.

75

75 **kODErrStorageUnitLocked**
Attempt to lock a storage unit that is already locked.

76

76 **kODErrInvalidStorageUnitKey**
Attempt to lock or unlock a storage unit with an invalid key.

77

77 **kODErrStorageUnitNotLocked**
Attempt to unlock a storage unit that is not locked.

84

84 **kODErrCannotCreateContainer**
Failure to create a container because its specified type is invalid.

85

85 **kODErrCannotOpenContainer**
Failure to open the physical container.

87

87 **kODErrNotImplemented**
The called method has not been implemented.

88

88 kODerrIteratorOutOfSync

This iterator is invalid because its list was changed after the iterator was created.

89

89 kODerrFocusAlreadyRegistered

The specified focus has already been registered.

90

90 kODerrFocusNotRegistered

The requested focus is not registered.

92

92 kODerrFocusIsNonExclusive

The focus has more than one owner.

95

95 kODerrCannotGetExternalLink

A link specification cannot create an external link.

96

96 kODerrCannotCreateLink

Failure to create the requested link source or its companion link object.

97

97 **kODErrNoLinkSpecValue**

The focused property does not contain a link-specification value.

98

98 **kODErrUnknownLinkSpecVersion**

The version of the link specifier is unknown.

99

99 **kODErrInvalidCanvas**

The specified canvas is not valid.

100

100 **kODErrCorruptLinkSpecValue**

The focused storage unit contains an invalid link-specification value.

101

101 **kODErrInvalidFacet**

The specified facet is not valid in the context in which it is used. For example, it is not the child or parent of the facet whose method is being called.

102

102 **kODErrUnsupportedFramePositionCode**

The frame position code specified for a new facet is not recognized.

104

104 **kODErrReadErr**

An error occurred while reading a storage unit value.

105

105 **kODErrWriteErr**

An error occurred while writing to a file or value.

106

106 **kODErrNoDragManager**

No platform-specific drag manager is available.

107

107 **kODErrNoDragSystemStorage**

The drag-and-drop object does not have system storage.

108

108 **kODErrDragItemNotFound**

The drag-and-drop object cannot find the item to be dragged.

109

109 kODErrCannotAllocateDragItem
The drag-and-drop object cannot allocate storage for the item to be dragged.

110

110 kODErrUnknownDragImageType
The drag-and-drop object does not recognize the specified drag-image type.

111

111 kODErrDragTrackingException
An exception occurred in the system-wide mouse tracking service.

112

112 kODErrNoShapeGeometry
A shape that is being used as a polygon lacks geometric information (its polygonal representation).

113

113 kODErrNotExportedLink
The specified link is not exported.

114

114 kODErrNotImportedLink
The link source is not imported.

115

115 kODerrCannotAcquireLink

Failure to recreate the link source or link object from the specified storage unit or link specification.

117

117 kODerrIllegalNullTransformInput

A frame method was passed null for a transform parameter that can not be null.

118

118 kODerrInvalidLinkKey

The specified link key is not valid.

119

119 kODerrDocNotSaved

A link could not be created because the source document has never been saved.

120

120 kODerrCannotMarkAction

Failure to start a subhistory by placing a mark at the beginning of the Undo and Redo Stacks; the undo object was never initialized properly.

121

121 kODerrEmptyStack

The Undo or Redo Stack is empty; the undo object was never initialized properly.

122

122 **kODErrNoBeginAction**

The undo object cannot find the begin action for this end action.

123

123 **kODErrCannotAddAction**

Attempt to add an action to the undo stack while already in the process of an undo or redo operation.

126

126 **kODErrInconsistentCloneKind**

The specified clone kind is used inconsistently. For example, a paste or drop clone kind can occur only following a copy or cut operation.

128

128 **kODErrTransformErr**

Attempt to perform an illegal operation on a transform object.

129

129 **kODErrInvalidParameter**

An invalid parameter was passed to a method.

131

131 **kODErrSubClassResponsibility**

The called method should have been overridden by the subclass of the class that defines the method, but was not.

132

132 **kODErrIllegalNullInput**

A method was passed null for a parameter that can not be null.

133

133 **kODErrObjectNotInitialized**

An object has not properly initialized.

135

135 **kODErrBrokenLink**

Internal error; the link source disconnected from its destinations.

136

136 **kODErrBrokenLinkSource**

The link has been broken at the source.

137

137 **kODErrPMEError**

An error has occurred in a presentation manager (PM) call.

138

138 kODerrInvalidObjectType

The specified object type is not valid in the context in which it is used.

139

139 kODerrInvalidPersistentObjectID

The specified identifier for a persistent object is not valid.

140

140 kODerrIllegalNullStorageSystemInput

A frame method was passed null for the storage system parameter.

141

141 kODerrNoEditionManager

The link manager does not have an edition file.

142

142 kODerrCannotRevealLink

Failure to show border around linked content.

143

143 kODerrCorruptLink

A link object is corrupt.

144

144 **kODerrLinkAlreadyExported**
The specified link has already been exported.

145

145 **kODerrCorruptLinkSource**
A link source is corrupt.

146

146 **kODerrCannotFindLinkSourceEdition**
Failure to locate the source of a cross-document link because the edition file does not exist.

147

147 **kODerrCannotFindLinkSource**
Failure to locate the source of a cross-document link.

148

148 **kODerrAlreadyImportedLink**
Failure to create a link due to an internal error.

157

157 **kODerrNullFacetInput**
Data interchange failed because a specified facet was null.

160

160 **kODErrRefCountNotEqualOne**

Attempt to delete a global reference-counted object while it is being used. (When a global object is not being used, only the session object has a reference to it, so its reference count is one.)

161

161 **kODErrInvalidDraftKey**

The specified draft key is not the draft key for the current cloning transaction.

162

162 **kODErrInvalidPersistentObject**

The specified persistent object is not valid.

163

163 **kODErrCloningInProgress**

164

164 **kODErrNoDocumentProperties**

165

165 **kODErrInvalidDestinationDraft**

The specified destination draft is invalid for the specified clone kind.

166

166 **kODErrDoesNotUndo**
This part does not support undo/redo.

167

167 **kODErrNoPromises**
This part does not fulfill promises.

168

168 **kODErrDoesNotDrop**
This part does not support drag and drop.

169

169 **kODErrDoesNotLink**
This part does not support linking.

171

171 **kODErrExistingDispatchModule**

172

172 **kODErrDispatcherNotInitialized**

173

173 **kODError**

174

174 **kODErrLinkSourceIsEmpty**

The link source was created but not initiated.

177

177 **kODErrInvalidTargetID**

An invalid connection was specified.

178

178 **kODErrRequestMutexTimeout**

The system timed-out while requesting a mutex semaphore.

179

179 **kODErrRequestMutexInterrupt**

The system was interrupted while requesting a mutex semaphore.

180

180 **kODErrRequestMutexInvalidHandle**

181

181 **kODErrRequestMutexTooManyRequests**

The maximum number of requests for this mutex has been exceeded.

182

182 **kODErrRequestMutexOwnerDied**

Attempt to begin a cloning transaction while another cloning transaction is in progress for the same draft.

183

183 **kODErrReleaseMutexInvalidHandle**

The specified handle of the requested mutex semaphore is not the owner.

184

184 **kODErrReleaseMutexNotOwner**

The caller requesting to release the mutex semaphore is not the owner.

185

185 **kODErrDocNotRegistered**

The document is not registered with the availability server. The document has not informed the availability server that it is open.

186

186 **kODErrDocNotOpened**

The document has not informed the availability server that is it open.

187

187 **kODErrLinkNotRegistered**

188

188 **kODErrLinkNotOpened**

189

189 **kODErrInvalidDocPathName**

The specified path name of the document does not match the one specified at registration time.

190

190 **kODErrDocAlreadyOpened**

The specified document is already opened.

191

191 **kODErrInvalidDocID**

The specified document ID is not associated with a registered document.

192

192 kODerrInvalidLinkID
An invalid link ID was specified.

193

193 kODerrInvalidAviSvr
The availability server cannot be created.

194

194 kODerrInvalidLinkSource
An invalid pointer to a link source was specified.

195

195 kODerrInvalidAuxState
An invalid pointer to an auxiliary state was specified.

196

196 kODerrLinkIDNotInDoc
The specified link ID cannot be found in this document.

197

197 kODerrInvalidShell

An invalid pointer to the availability server shell was specified.

198

198 **kODErrLinkAlreadyOpened**

The availability server has already been informed that this link is open.

199

199 **kODErrWaitingUnlocked**

The link source is expecting to be unlocked.

202

202 **kODErrInvalidLink**

An invalid pointer to a link was specified.

223

223 **kODMaxLinkError**

230

230 **kODErrNoWindow**

A window state's [RegisterWindow](#) method was passed null for the platform window parameter.

231

231 **kODErrInvalidCloneKind**
The specified clone kind is not valid.

171

232 **kODErrIllegalNullTokenInput**
A frame method was passed null for a token parameter that can not be null.

233

233 **kODErrIllegalNullShapeInput**
A frame method was passed null for the shape parameter that can not be null.

234

234 **kODPartNotWrapper**

235

235 **kODErrIllegalNullStorageUnitInput**
A method was passed null for a storage unit parameter that can not be null.

241

241 **kODErrInvalidExtension**
The specified extension object is not valid in the context in which it is used.

242

242 **kODerrUnknownExtension**
The specified extension is not a known extension.

243

243 **kODerrIteratorNotInitialized**
A method was called on an uninitialized iterator.

181

244 **kODerrIllegalNullFacetInput**
A frame method was passed null for the facet parameter that cannot be null.

245

245 **kODerrIllegalNullFrameInput**
A frame method was passed null for the frame parameter that can not be null.

246

246 **kODerrValueIndexOutOfRange**
The specified property has no value at the specified index.

247

247 **kODerrMoveIntoSelf**
A clone operation attempted to move a part such that it would become embedded within itself.

248

248 **kODErrNullLinkInfoInput**

The specified parameter of type [ODLinkInfo fomr=textonly](#) is null.

249

249 **kODErrNullLinkInfoResultInput**

The specified parameter of type [ODLinkInfoResult fomr=textonly](#) is null.

250

250 **kODErrNullPasteAsResultInput**

251

251 **kODErrNullDestinationFrame**

The *destFrame* parameter to the [BeginClone](#) method is null.

253

253 **kODErrUnknownUpdateID**

The specified update ID is the reserved value kODUnknownUpdate.

254

254 **kODErrNoLinkContent**

The link contents storage unit has no contents property.

255

255 kODErrNotRootFrame
This frame is not the root frame.

256

256 kODErrIllegalRecursiveEmbedding
Attempt to embed a frame in its own part or in a containing part of its part.

257

257 kODErrIllegalClipboardCloneKind

257

257 kODMaxError
The maximum numerical value for error codes.

Notices

Edition Notices

August 1996

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM reseller or IBM marketing representative.

Copyright Notices

(C) Copyright International Business Machines Corporation 1996. All rights reserved.

(C) Copyright Apple Computer, Inc 1995. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Disclaimers

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Asia-Pacific users can inquire, in writing, to the IBM Director of Intellectual Property and Licensing, IBM World Trade Asia Corporation, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106, Japan.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Department LZKS, 11400 Burnet Road, Austin, TX 78758 U.S.A. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
Common User Access
CUA
IBM
Operating System/2
OS/2
Personal System/2
Presentation Manager
SAA
System/38
Systems Application Architecture
WIN-OS/2
Workplace Shell

The following terms are trademarks of other companies:

Adobe	Adobe Systems Incorporated
Apple	Apple Computer, Inc.
C++	American Telephone and Telegraph Company
CORBA	Object Management Group, Inc.
Display PostScript	Object Management Group, Inc.
Hewlett-Packard	Hewlett-Packard Company
HP	Hewlett-Packard Company
LaserJet	Hewlett-Packard Company
Macintosh	Apple Computer, Inc.
Microsoft	Microsoft Corporation
OpenDoc	Apple Computer, Inc.
PostScript	Adobe Systems Incorporated
QuickDraw	Apple Computer, Inc.
Windows	Microsoft Corporation

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.
